GE
Security

User Manual

Picture Perfect v4 External Interface

Picture Perfect 4.0 External Interface

User Manual

Copyright Copyright © 2006, GE Security Inc. All rights reserved.

This document may not be copied or otherwise reproduced, in whole or in part, except as specifically permitted under US and international copyright law, without the prior written consent from GE.

Document number/460588003A (March 2006).

Disclaimer THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE. GE ASSUMES NO RESPONSIBILITY FOR INACCURACIES OR OMISSIONS AND SPECIFICALLY DISCLAIMS ANY LIABILITIES, LOSSES, OR RISKS, PERSONAL OR OTHERWISE, INCURRED AS A CONSEQUENCE, DIRECTLY OR INDIRECTLY, OF THE USE OR APPLICATION OF ANY OF THE CONTENTS OF THIS DOCUMENT. FOR THE LATEST DOCUMENTATION, CONTACT YOUR LOCAL SUPPLIER OR VISIT US ONLINE AT WWW.GESECURITY.COM.

This publication may contain examples of screen captures and reports used in daily operations. Examples may include fictitious names of individuals and companies. Any similarity to names and addresses of actual businesses or persons is entirely coincidental.

Trademarks and patents GE and the GE monogram are registered trademarks of General Electric.
Picture Perfect product and logo are trademarks of GE Security.

Other trade names used in this document may be trademarks or registered trademarks of the manufacturers or vendors of the respective products.

Intended use Use this product only for the purpose it was designed for; refer to the data sheet and user documentation. For the latest product information, contact your local supplier or visit us online at `www.gesecurity.com`.

Contents

Preface . . . . . **vii**
Conventions used in this document vii
Safety terms and symbols vii

Chapter 1. Configuring EIF 1

Overview **2**
External Interface Programs **3**
Picture Perfect Event Manager 3
Picture Perfect Request Manager 3
Configuration **4**
Event Manager Configuration File 4
Request Manager Configuration File 5
Diagnostics **7**
Verifying EIF is running 7
Verifying sessions are active 7
Sessions 8
Session Life-cycle 8

Chapter 2. Using EIF 11

Overview **12**
Naming Conventions **12**
XML-RPC Data Types and Parameter Parsing **13**
EIF Events **13**
Event Protocol 14
Ping Event (hostname.ping) 14
Alarm Event (hostname.processAlarm) 15
Alarm Process State Change Event (hostname.processAlarmProcessState) 17
Alarm Refresh Event Array (hostname.processAlarmRefresh) 18
Badge Event (hostname.processBadge) 21
Input Event (hostname.processInputActivity) 24
Operator Event (hostname.processOperActivity) 24
Popup Event (hostname.processPopup) 25
PPRS Mode Change Event (hostname.processPPRSModeChange) 26
Single Alarm Refresh Event (hostname.processSingleAlarmRefresh) 26
Tour Event (hostname.processTourActivity) 29
Event Response Object 30
EIF Commands **30**
Command Protocol 31
Activate Output (picture.activateOutput) 31
Force Alarm Refresh (picture.alarmRefresh) 32
Change Mode (picture.changeMode) 33
Check or Add Port (picture.checkAddPort) 34
Disable Input Group on Micro (picture.disableIngrp) 34
Disable Input (picture.disableInput) 35
Enable Input Group (picture.enableIngrp) 36
Enable Input (picture.enableInput) 36
Lock Door (picture.lockDoor) 37
Logout (picture.logout) 38
Create Operator History (picture.operatorHistory) 39
Change Modes (picture.ppChangeMode) 41
Validate Printer Queue (picture.printerqueue) 42
Reset Micro (picture.resetMicro) 43
Reset Output (picture.resetOutput) 44
Reset Timed APB (picture.resetTimedAPB) 45
Set Diagnostics (picture.setDiagnostics) 46
Set Ping Interval (picture.setPingInterval) 48
Tour Functions (picture.tourFunctions) 49
TTY Validate (picture.ttyValidate) 50
Unlock Door (picture.unlockDoor) 51
Process Alarm (picture.updateAlarmProcessingState) 52
EIF Data Requests **53**
EIF Data Requests Method Table 53
Request Protocol 54

Convert Date and Time (picture.convertDateTime) 54
Directory Listing (picture.directoryListing) 55
Get Alarm Status (picture.getAlarmStatus) 56
Get Configured Devices (picture.getConfiguredDevices) 57
Get Diagnostics (picture.getDiagnostics) 58
Get History Counts (picture.getHistoryCounts) 60
Get History Flags (picture.getHistoryFlags) 61
Get Host ID (picture.getHostId) 62
Get License Information (picture.getLicenseInformation) 63
Get Micro Time (picture.microTime) 66
Get Operators Logged In (picture.getOperatorsLoggedIn) 67
Get Host Time (picture.hostTime) 68
Login (picture.loginEIF) 69
Query Access Points (picture.queryAccessPoints) 72
Query Alarm Instructions (picture.queryAlarmInstructions) 73
Query Alarm Responses (picture.queryAlarmResponses) 74
Query Facilities (picture.queryFacilities) 75
Query Input Groups (picture.queryInputGroups) 76
Query Inputs (picture.queryInputs) 77
Query Micro State (picture.queryMicroState) 78
Query Modes (picture.queryModes) 81
Query Operators (picture.queryOperators) 82
Query Outputs (picture.queryOutputs) 83
Query Port State (picture.queryPortState) 84
Query Status (picture.queryStatus) 86

Chapter 3. Examples 87

Overview **88**
Java Implementation Examples **88**
Sample EIF XML-RPC Client in Java 88
Sample EIF XML-RPC Server in Java 91
Perl Implementation Examples **93**
Sample EIF XML-RPC Client in Perl 93
Sample EIF XML-RPC Server in Perl 95

Chapter 4. Contacting technical support 97

Contacting technical support **98**
Online publication library **98**

Index 99

Preface

This manual is intended for systems integrators and/or network programmers seeking methods for integrating into the Picture Perfect platform using XML-RPC as the communications protocol.

This manual provides information for a knowledgeable XML-RPC programmer to interface external programs to the Picture Perfect system. It is not intended as a tutorial on XML-RPC programming. If you need additional information on XML-RPC, please contact your GE Customer Support Representative for a list of recommended sources. For additional information on XML-RPC, please visit *http://www.xmlrpc.com* on the web.

Read these instructions and all ancillary documentation entirely before installing or operating this product. The most current versions of this and related documentation may be found on our website. Refer to *Online publication library* on page 98 for instructions on accessing our online publication library.

Note: A qualified service person, complying with all applicable codes, should perform all required hardware installation.

Conventions used in this document

The following conventions are used in this document:

Bold	Menu items and buttons.
Italic	Emphasis of an instruction or point; special terms.
	File names, path names, windows, panes, tabs, fields, variables, and other GUI elements.
	Titles of books and various documents.
Blue italic	(Electronic version.) Hyperlinks to cross-references, related topics, and URL addresses.
`Monospace`	Text that displays on the computer screen.
	Programming or coding sequences.

Safety terms and symbols

These terms may appear in this manual:

CAUTION: *Cautions* identify conditions or practices that may result in damage to the equipment or other property.

WARNING: *Warnings* identify conditions or practices that could result in equipment damage or serious personal injury.

Chapter 1 Configuring EIF

This section covers the information needed to configure the Picture Perfect External Interface software.

In this chapter:

Overview 2
External Interface Programs 3
Configuration 4
Diagnostics 7

Overview

The Picture Perfect External Interface (EIF) is a package that enables the publishing of events to remote clients, remote access to native access control functions, and remote access to data stored in the Informix database. This provides an open and standardized way of integrating Picture Perfect access control systems with external systems via XML-RPC. The EIF package effectively exposes Picture Perfect functionality in the form of an XML object that can be manipulated using XML-RPC.

When using the EIF package, a great deal of Picture Perfect functionality is immediately exposed for manipulation by systems integrators. This functionality can logically be divided into three areas: Events, Commands, and Data Requests. Events are the means in which an EIF client can receive Picture Perfect badge and alarm activity. Commands are the means in which an EIF client can manipulate Picture Perfect access control functions such as the locking/unlocking of doors, activation of inputs/outputs, and remote responses to alarms. Data Requests provide the means by which an EIF client can obtain database records for hosts, operators, facilities, access points, and others. For a complete listing of all Events, Command, and Data Requests that EIF provides please refer to *Chapter 2 Using EIF*.

In order to enable communication with EIF clients, the EIF package provides two Picture Perfect daemons to handle Events, Commands, and Data Requests. These daemons are the Picture Perfect Event Manager, which is responsible for handling all outbound Events, and the Picture Perfect Request Manager, which is responsible for handling all incoming Commands and Data Requests. These daemons communicate with all EIF clients using XML-RPC to transmit Events, Commands, and Data Requests as well as any necessary responses. All communications between EIF clients and the Picture Perfect system enforce user authentication. Optionally, the use of SSL encryption for communications can be set up to provide an additional layer of security.

External Interface Programs

The Picture Perfect External Interface is primarily comprised of two programs, an Event Manager daemon that publishes Events to an EIF client and a Request Manager daemon that services Commands and Data Requests from an EIF client. These daemons are spawned by TPS and listed in the tps_daemons table.

Picture Perfect Event Manager

The Picture Perfect Event Manager is a UNIX daemon spawned by TPS. It is responsible for receiving Events from various Picture Perfect processes and publishing them to the EIF client. The Event Manager handles all EIF Events, including Badge and Alarm Events.

Picture Perfect Request Manager

The Picture Perfect Request Manager is a UNIX daemon spawned by TPS. It is responsible for receiving Commands and Data Requests from EIF clients and returning the appropriate result set. The Request Manager handles all EIF Commands, including Login and Logout. It also handles all EIF Data Requests, including Query Operators and Query Facilities.

Configuration

Two files are used to configure the communications between the Event/Request Managers and the EIF client. These files are located in **`/cas/db/text`**. There is a configuration file for the Event Manager (**`eifevtmgr.cfg`**) and one for the Request Manager (**`eifreqmgr.cfg`**). These files contain configuration parameters such as port numbers used for communicating with EIF clients.

It is important to note that if you change any parameters, then Picture Perfect must be restarted. Reminder:

Table 1. Starting and Stopping Picture Perfect

Picture Perfect System	Stop	Start
Standalone and Enterprise editions	`rc.pperf -k`	`rc.pperf`
Redundant	`pprscmd stop`	`pprscmd start`

Event Manager Configuration File

eifevtmgr.cfg

Type the following command:
`cat /cas/db/text/eifevtmgr.cfg` Enter

and output similar to the following displays:

```
#
# Copyright (C) 2002-2004 GE Security
# All Rights Reserved.
#
# eifevtmgr.cfg
#
# @(#) eifevtmgr.cfg 1.0 12/12/02
#
# Supported Property Examples
#
# URL=http://hostname/FacilityCommander/servlet/
# net.casi.app.AppHttpServlet
# PORT=8085
# KEEP_ALIVE=TRUE
# MESSAGE_TRACE=FALSE
# RESEND_BUFFER_SIZE=256
# Default SSL port
# PORT=8443
# KEEP_ALIVE=FALSE
# MESSAGE_TRACE=TRUE
# RESEND_BUFFER_SIZE=256
# RANDOM=/frozen/pp1.70/cas/src/eif/random.pem
```

```
# CERTIFICATE=/cas/db/text/tmcclure.cer
#
URL=http://pphost/
PORT=8085
KEEP_ALIVE=TRUE
```

Table 2. eifevtmgr.cfg

Parameter	Definition	Required	Default
PORT	Defines the port on which the **eifevtmgr** listens.	Yes	Define a single port or a range, such as 8085-8100
KEEP_ALIVE	Indicates whether the socket connection is closed immediately after receiving a request on the listening socket.	No	True
CLIENT_AUTH	True or False turns client authentication on or off.	No	False
MESSAGE_TRACE	Causes the **eifevtmgr** to log the contents of each XML message sent to the EIF client. This should be used with caution as the messages are verbose and may fill up the log if left on for an extended period of time.	No	False

Request Manager Configuration File

eifreqmgr.cfg

Type the following command:
`cat /cas/db/text/eifreqmgr.cfg` Enter
and output similar to the following displays:

```
# Copyright (C) 2002-2004 GE Security
# All Rights Reserved.
#
# eifreqmgr.cfg
#
# @(#) eifreqmgr.cfg 1.0 12/12/02
#
# Supported Property Examples
#
# Non-SSL connection
# PROTOCOL=HTTP
# SSL connection
# PROTOCOL=HTTPS
# PORT=8088
# KEEP_ALIVE=TRUE
# SESSION_TIMEOUT=10
# MESSAGE_TRACE=TRUE
```

```
# RANDOM=/frozen/pp1.70/cas/src/eif/random.pem
# KEYFILE=/cas/db/text/facilitycommanderaix.pem
#
PROTOCOL=HTTP
PORT=8088
KEEP_ALIVE=TRUE
```

Table 3. eifreqmgr.cfg

Parameter	Definition	Required	Default
PROTOCOL	Designates either HTTP or HTTPS (secured) protocol.	Yes	HTTP
PORT	Defines the port on which the **eifreqmgr** listens.	Yes	8043
KEEP_ALIVE	Indicates whether the socket connection is closed immediately after receiving a request on the listening socket.	No	True
MESSAGE_TRACE	Causes the **eifreqmgr** to log the contents of each XML message sent to the EIF client. This should be used with caution as the messages are verbose and may fill up the log if left on for an extended period of time.	No	False

Diagnostics

When attempting to diagnose or debug issues with the EIF package, there are two levels of diagnostics available to the systems integrator. First, Picture Perfect supports diagnostic logging through the `setdiag` utility command. There are two flags that support each of the new EIF Programs:

Table 4. Diagnostics

Action	Command
Turn on diags for eifevtmgr	setdiag +eifevtmg
Turn on diags for eifreqmgr	setdiag +eifreqmg
Turn off diags for eifevtmgr	setdiag -eifevtmg
Turn off diags for eifreqmgr	setdiag -eifreqmg

Second, the EIF Event and Request Managers configuration files support a parameter called MESSAGE_TRACE. When this is set to True, the Picture Perfect log file `/cas/log/log.MMDD` will contain the complete text of each XML-RPC message being transmitted and received by the EIF Event and Request Managers. This can be helpful when diagnosing XML-RPC message syntax issues with EIF Events, Commands, and Requests.

Verifying EIF is running

When the system is running, log in as **`root`** and verify that EIF is running by typing at the **`#`** prompt:

```
ps -ef | grep eif Enter
```

You should get output similar to the following:

```
root 12579 17148 0 10:19:36 pts/0 0:00 eifevtmgr
root 12500 17148 0 10:19:36 pts/0 0:00 eifreqmgr
```

Verifying sessions are active

There is a utility that can be used to list the active sessions that currently exist on a running Picture Perfect system. This utility is called **`smutl`**. When the system is running, log in as **`root`** and run **`smutl`** as follows by typing at the **`#`** prompt:

```
smutl –o -1 Enter
```

You should get output similar to the following for each session currently logged in:

```
Dump all operators
==========================================
operator[ 0].session:C3EALQF421
```

```
operator[ 0].user_id:install
operator[ 0].address:3.112.49.75
operator[ 0].port:8090
operator[ 0].id:1
operator[ 0].time_zone:0
operator[ 0].login.date:12/21/2004
operator[ 0].login.time:15:53:33
operator[ 0].terminal:0
operator[ 0].facility_list:
Total facilities:0

Number of operators currently in shared memory ......: 1
```

Sessions

All External Interface interactions between a client and a Picture Perfect host happen within the context of an active session. Having an active session is a prerequisite to using EIF Commands, Data Requests, and Events properly. Therefore, before attempting to utilize the EIF API, it is first necessary to understand the creation, longevity, and the removal of active sessions.

Session Life-cycle

The only way to create an active session is by using the **`picture.loginEIF`** EIF Request with valid operator information and a unique SESSION ID. Please be aware of the following restrictions on SESSION ID creation:

Table 5. Session ID Restrictions

Restrictions on SESSION ID
The SESSION ID cannot exceed 128 characters in length.
The SESSION ID must contain only alphanumeric characters.

An active session exists as soon as the **`picture.loginEIF`** EIF Request returns a RESULT of TRUE. The session will remain active from that point until either the Picture Perfect host is shutdown or the `picture.logout` EIF Command is executed with the same SESSION ID, whichever happens first.

Once the SESSION ID has been used to create an active session, all subsequent EIF Commands and Data Requests can be invoked by passing the SESSION ID that is currently active. If an invalid SESSION ID is provided, then each EIF Command or Data Request will fail with a RESULT of FALSE.

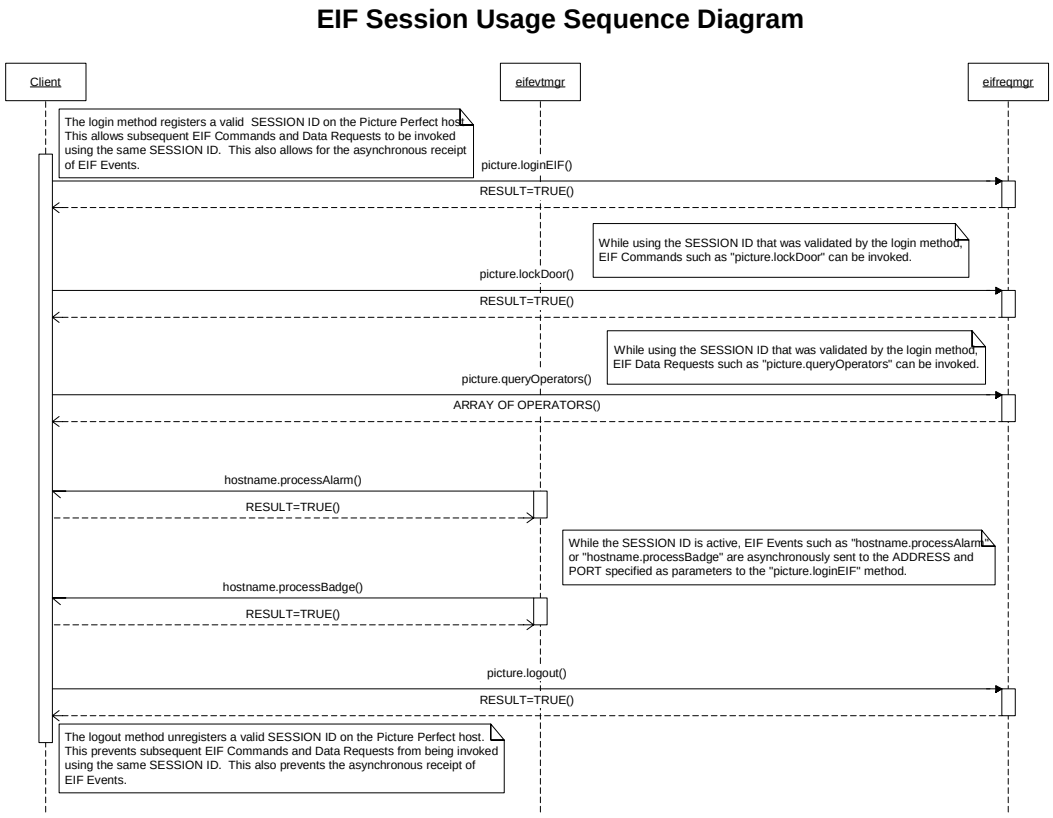
In addition to having the ability to invoke EIF Commands and Data Requests, the existence of an active session allows for the receipt of real-time asynchronous EIF Events such as those that represent alarm and badge transactions that occur in a Picture Perfect system. The EIF Events will always be transmitted via XML-RPC to the IP address and port number specified by the ADDRESS and PORT parameters of the **`picture.loginEIF`** EIF Request that created the active session.

Note: The EIF Events transmitted to a client have already had routing rules applied to them. A client will only receive EIF Events if they have at least been setup to be routed to MONITOR. It is the responsibility of the client to enforce any Facility filtering that is desired. By design EIF Events from all facilities are transmitted to the client.

When a session is no longer needed, the **`picture.logout`** EIF Command can be used to terminate an active session. Afterwards, subsequent EIF Commands and Data Requests will no longer be possible using the terminated SESSION ID. Also, EIF Events will no longer be transmitted to the client.

The following diagram displays an example sequence of events required create an active session, exercise EIF Commands, Data Requests, and Events, and terminate an active session

Figure 1. EIF Session Usage Sequence Diagram.


EIF Session Usage Sequence Diagram
Client
eifevtmgr
eifreqmgr
The login method registers a valid SESSION ID on the Picture Perfect host. This allows subsequent EIF Commands and Data Requests to be invoked using the same SESSION ID. This also allows for the asynchronous receipt of EIF Events.
picture.loginEIF()
RESULT=TRUE()
While using the SESSION ID that was validated by the login method, EIF Commands such as "picture.lockDoor" can be invoked.
picture.lockDoor()
RESULT=TRUE()
While using the SESSION ID that was validated by the login method, EIF Data Requests such as "picture.queryOperators" can be invoked.
picture.queryOperators()
ARRAY OF OPERATORS()
hostname.processAlarm()
RESULT=TRUE()
While the SESSION ID is active, EIF Events such as "hostname.processAlarm" or "hostname.processBadge" are asynchronously sent to the ADDRESS and PORT specified as parameters to the "picture.loginEIF" method.
hostname.processBadge()
RESULT=TRUE()
picture.logout()
RESULT=TRUE()
The logout method unregisters a valid SESSION ID on the Picture Perfect host. This prevents subsequent EIF Commands and Data Requests from being invoked using the same SESSION ID. This also prevents the asynchronous receipt of EIF Events.

Chapter 2 Using EIF

This section covers the protocol and XML-RPC tag-based formatting of each Picture Perfect EIF Event, Command, and Data Request.

In this chapter:

Overview 12
Naming Conventions 12
XML-RPC Data Types and Parameter Parsing 13
EIF Events 13
EIF Commands 30
EIF Data Requests 53

Overview

The Picture Perfect External Interface relies on XML-RPC as the protocol for information transport between Picture Perfect and EIF clients. XML-RPC is a specification and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet. XML-RPC is basically remote procedure calling using HTTP as the transport and XML as the encoding. It is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed, and returned. For further information regarding XML-RPC please visit: http://www.xmlrpc.com

The Picture Perfect External Interface application-programming interface (API) is divided into three areas: Events, Commands, and Data Requests. In this chapter you will find the protocol and XML-RPC tag-based formatting of each Picture Perfect EIF Event, Command, and Data Request.

Naming Conventions

When invoking/receiving EIF Events, Commands, or Data Requests, the Picture Perfect EIF package uses the following naming conventions.

To receive EIF Events:

The object name **`hostname`** is always the name of the host running your EIF Server, not necessarily the Picture Perfect server. The hostname is always referenced in the URL parameter of **`/cas/db/text/eifevtmgr.cfg`**.

When invoking EIF Events, the Picture Perfect EIF package uses the following naming conventions.

```
<methodCall>
<methodName>hostname.method</methodName>
…
</methodCall>
```

For example, when receiving an EIF Alarm Event, the methodName would be:

```
<methodCall>
<methodName>hostname.processAlarm</methodName>
…
</methodCall>
```

EIF Commands or Data Requests:

The object name **`picture`** is always referenced when invoking EIF Command or Data Request methods. Do not use any object name other than **`picture`** for handling EIF Commands or Data Requests.

When invoking EIF Commands or Data Requests, the Picture Perfect EIF package uses the following naming conventions.

```
<methodCall>
<methodName>picture.method</methodName>
…
</methodCall>
```

For example, to invoke the EIF Login command, the methodName would be:

```
<methodCall>
<methodName>picture.loginEIF</methodName>
…
</methodCall>
```

XML-RPC Data Types and Parameter Parsing

The following data types are supported by XML-RPC and used by the Picture Perfect External Interface:

Table 6. XML-RPC Data Types

Data Type	Description
int	A signed, 32-bit integer.
string	An ASCII string that may contain NULL bytes.
boolean	Either true or false.
double	A double-precision floating point number. (Accuracy may be limited in some implementations.)
base64	Raw binary data of any length; encoded using Base64.
array	A one-dimensional array of values. Individual values may be of any type.
struct	A collection of key-value pairs. The keys are strings; the values may be of any type.

It is important to note that various XML-RPC packages handle method parameters in different ways. In Java and Perl, the XmlRpc and Frontier-RPC packages respectively parse method parameters into hash tables that are keyed by parameter name to determine the parameter value. Some data types such as boolean and base64 may not be automatically translated into a native type in a particular programming language. Please refer to your XML-RPC package documentation for information regarding the handling and parsing of method parameters.

EIF Events

This section describes the EIF Events supported by the API. The name of the Event and all of its fields are listed.

Table 7. EIF Events Method

Event Name	Method Name	Event Description
Ping Event	hostname.ping	Object containing information concerning the state of the Picture Perfect system and EIF version.
Alarm Event	hostname.processAlarm	Object containing all information concerning an alarm transaction that has occurred.
Alarm Process State Change Event	hostname.processAlarmProcessState	Object containing all information concerning the change in state of an alarm, including the operator's response and the alarm state.
Alarm Refresh Event Array	hostname.processAlarmRefresh	An array of objects containing all information concerning every existing alarm in the Picture Perfect alarm buffer.
Badge Event	hostname.processBadge	Object containing all information concerning a badge transaction that has occurred.
Event Activity Event	hostname.processEventActivity	Object containing all information concerning an event transaction that has occurred.

Table 7. EIF Events Method (continued)

Event Name	Method Name	Event Description
Input Activity Event	hostname.processInputActivity	Object containing all information concerning an input transaction that has occurred.
Operator Activity Event	hostname.processOperActivity	Object containing all information concerning an operator transaction that has occurred.
Popup Event	hostname.processPopup	Object containing the text of informational popup messages that are displayed to operators when "Receive System Notifications" is enabled.
PPRS Mode Change Event	hostname.processPPRSModeChange	Event that is sent when there is a change in Primary/ Backup state of a server in a Redundant system.
Single Alarm Refresh Event	hostname.processSingleAlarmRefresh	Object containing information concerning a single alarm in the Picture Perfect alarm buffer. Only transmitted when an operator changes the alarm from not being routed to being routed.
Tour Activity Event	hostname.processTourActivity	Object containing all information concerning a tour transaction that has occurred.

Event Protocol

The protocol for properly receiving Events from an EIF client requires that upon receipt of each Event, the EIF client must transmit an Event Response back to Picture Perfect External Interface. This is necessary to avoid the resending of Events to the EIF client.

Ping Event (hostname.ping)

The Ping Event Object contains information concerning the state of the Picture Perfect system and EIF version.

Table 8. Ping Event

Field Name	Field Description	Data Type
HOST_ID	ID of the Picture Perfect host.	int
PP_REFRESH	Flag if the alarms need to be refreshed due to a communication failure or restart.	boolean
TPS_MODE1	In a Redundant system, the Primary installed machine's tps mode: 0=Standalone 1=Primary 2=Backup	int
TPS_MODE2	In a Redundant system, the Backup installed machine's tps mode: 0=Standalone 1=Primary 2=Backup	int
VERSION	Interface version.	string

Ping Example Message

```
<methodCall>
 <methodName>ppmerlin.ping</methodName>
 <params>
  <param><value><struct>
   <member><name>PP_REFRESH</name><value><boolean>0</boolean></value></member>
   <member><name>VERSION</name><value><string>v0.5.3</string></value></member>
   <member><name>HOST_ID</name><value><int>1</int></value></member>
   <member><name>TPS_MODE1</name><value><int>0</int></value></member>
  <member><name>TPS_MODE2</name><value><int>0</int></value></member>
  </struct></value></param>
 </params>
</methodCall>
```

Alarm Event (hostname.processAlarm)

The Alarm Event Object contains all of the information relevant to an alarm transaction that has occurred on the Picture Perfect system.

Table 9. Alarm Event

Field Name	Field Description	Data Type
ALARM_DESC	Description of the alarm.	string
ALARM_ID	ID of the alarm.	int
DATE	Date of the alarm. Format in YYYYMMDD.	int
FACILITY_DESC	Description of the facility of the alarm location (not supported).	string
FACILITY_ID	ID of the facility of the alarm location.	int
HOST_DATE	Date of the alarm in HOST TIME. Format in YYYYMMDD.	int
HOST_DESC	Description of the host that processed the badge read. (not supported)	string
HOST_ID	ID of the host that processed the alarm.	int
HOST_TIME	Time of the alarm in HOST TIME. Format in HHMMSS.	int
INGRP_ID	ID of the alarm's input group.	int
LOC_DESC	Location description of the alarm point.	string
LOC_ID	Location ID of the alarm point.	int
LOGICAL_ST_DESC	Description of the alarm's logical state.	string
LOGICAL_ST_ID	ID indicating the alarm's logical state: 1=set 2=reset 3=tamper	int
PHYS_ST_ID	ID of the alarm's physical state: 1=short 2=cut 3=open 4=closed	int
PRIOR_LVL	Priority level of the alarm.	int

Table 9. Alarm Event (continued)

Field Name	Field Description	Data Type
PROC_ST_ID	ID of the alarm's process state: 0=active 1=bumped 2=notified 3=remote 4=pending 5=completed 6=removed 7=purged	int
REMOVE_SET	Flag that indicates if the alarm can be removed while in a set condition.	boolean
RESET_COUNT	Count of the number of times this alarm has been reset in the Picture Perfect alarm buffer.	int
SET_COUNT	Count of the number of times this alarm has been set in the Picture Perfect alarm buffer.	int
TABLE_ID	Table ID of the alarm location.	int
TIME	Time of the alarm. Format in HHMMSS.	int
TIMESTAMP	Event timestamp formatted in number of seconds since Jan 1 1970.	int
TZ	Time zone of the Picture Perfect system.	string
UTC_DATE	Date of the alarm in UTC time. Format in YYYYMMDD.	int
UTC_TIME	Time of the alarm in UTC time. Format in HHMMSS.	int

ALarm Event Example (XML-RPC)

```
<methodCall>
<methodName>ppmerlin.processAlarm</methodName>
 <params>
         <param><value><struct>
         <member><name>ALARM_ID</name><value><int>16</int></value></member>
         <member><name>ALARM_DESC</name><value><string>DOOR FORCED OPEN</string></value></member>
         <member><name>DATE</name><value><int>20030405</int></value></member>
         <member><name>HOST_DATE</name><value><int>20030405</int></value></member>
         <member><name>UTC_DATE</name><value><int>20030405</int></value></member>
         <member><name>TIME</name><value><int>134016</int></value></member>
         <member><name>HOST_TIME</name><value><int>134016</int></value></member>
         <member><name>UTC_TIME</name><value><int>134016</int></value></member>
         <member><name>TZ</name><value><string>EDT</string></value></member>
         <member><name>TABLE_ID</name><value><int>16</int></value></member>
         <member><name>PRIOR_LVL</name><value><int>10</int></value></member>
         <member><name>HOST_ID</name><value><int>1</int></value></member>
         <member><name>HOST_DESC</name><value><string>Host Name</string></value></member>
         <member><name>LOC_ID</name><value><int>1793</int></value></member>
    <member><name>LOC_ST_DESC</name><value><string>STATE</string></value></member>
         <member><name>LOC_DESC</name><value><string>160-1-0 DOOR</string></value></member>
         <member><name>INGRP_ID</name><value><int>27</int></value></member>
         <member><name>LOGICAL_ST_ID</name><value><int>1</int></value></member>
         <member><name>PHYS_ST_ID</name><value><int>3</int></value></member>
         <member><name>PROC_ST_ID</name><value><int>0</int></value></member>
         <member><name>FACILITY_ID</name><value><int>1</int></value></member>
         <member><name>FACILITY_DESC</name><value><string>FACILTY_DESC</string></value></member>
         <member><name>TIMESTAMP</name><value><int>1049568016</int></value></member>
         <member><name>SET-COUNT</name><value><int>9</int></value></member>
         <member><name>RESET_COUNT</name><value><int>9</int></value></member>
         </struct></value></param>
</params>
</methodCall>
```

Alarm Process State Change Event (hostname.processAlarmProcessState)

The Alarm Process State Change Event Object contains all of the information concerning the change in state of an alarm, including the operator's response and the alarm state.

Table 10. Alarm Process State Change Event

Field Name	Field Description	Data Type
ALARM_ID	ID of the alarm.	int
DATE	Date of alarm processing. Format in YYYYMMDD.	int
HOST_DATE	Date of the alarm processing in HOST TIME. Format in YYYYMMDD.	int
HOST_DESC	Description of the host that processed the alarm. (not supported)	string
HOST_ID	ID of the host that processed the alarm.	int
HOST_TIME	Time of the alarm processing in HOST TIME. Format in HHMMSS.	int
INGRP_ID	ID of the alarm's input group.	int
LOC_ID	Location ID of the alarm point.	int
PROC_ST_ID	ID of the alarm's process state: 0=active 1=bumped 2=notified (nan) 3=notified (ran) 4=pending 5=completed 6=removed 7=purged	int
RESPONSE	Operator response for process state change.	string
TABLE_ID	Table ID of the alarm location.	int
TIME	Time of the alarm processing. Format in HHMMSS.	int
TIMESTAMP	Event timestamp formatted in number of seconds since Jan 1 1970.	int
TZ	Time zone of the Picture Perfect system.	string
USER_ID	Login name of operator processing alarm.	string
UTC_DATE	Date of the alarm processing in UTC time. Format in YYYYMMDD.	int
UTC_TIME	Time of the alarm processing in UTC time. Format in HHMMSS.	int

Alarm Process State Change Event Example (XML-RPC)

```
<methodCall>
 <methodName>ppmerlin.processAlarmProcessState</methodName>
 <params>
  <param><value><struct>
   <member><name>ALARM_ID</name><value><int>16</int></value></member>
   <member><name>DATE</name><value><int>20030405</int></value></member>
   <member><name>HOST_DATE</name><value><int>20030405</int></value></member>
   <member><name>UTC_DATE</name><value><int>20030405</int></value></member>
   <member><name>TIME</name><value><int>140107</int></value></member>
   <member><name>HOST_TIME</name><value><int>140107</int></value></member>
```

```
   <member><name>UTC_TIME</name><value><int>140107</int></value></member>
   <member><name>TZ</name><value><string>EDT</string></value></member>
   <member><name>TABLE_ID</name><value><int>16</int></value></member>
   <member><name>HOST_ID</name><value><int>1</int></value></member>
   <member><name>INGRP_ID</name><value><int>27</int></value></member>
   <member><name>HOST_DESC</name><value><string>Host Name</string></value></member>
   <member><name>LOC_ID</name><value><int>1793</int></value></member>
   <member><name>PROC_ST_ID</name><value><int>4</int></value></member>
   <member><name>RESPONSE</name><value><string>LOCAL POLICE CALLED</string></value></member>
   <member><name>USER_ID</name><value><string>install</string></value></member>
   <member><name>TIMESTAMP</name><value><int>1049569267</int></value></member>
  </struct></value></param>
 </params>
</methodCall>
```

Alarm Refresh Event Array (hostname.processAlarmRefresh)

The Alarm Refresh Event Array Object is an array of objects containing all information concerning every existing alarm in the Picture Perfect alarm buffer.

Table 11. Alarm Refresh Event Array

Field Name	Field Description	Data Type
ALARM_DESC	Description of the alarm.	string
ALARM_ID	ID of the alarm.	int
DATE	Date of the alarm. Format in YYYYMMDD.	int
FACILITY_DESC	Description of the facility of the alarm location. (not supported)	string
FACILITY_ID	ID of the facility of the alarm location.	int
HOST_DATE	Date of the alarm in HOST TIME. Format in YYYYMMDD.	int
HOST_DESC	Description of the host that processed the badge read (not supported).	string
HOST_ID	ID of the host that processed the alarm.	int
HOST_TIME	Time of the alarm in HOST TIME. Format in HHMMSS.	int
INGRP_ID	ID of the alarm's input group.	int
LOC_DESC	Location description of the alarm point.	string
LOC_ID	Location ID of the alarm point.	int
LOGICAL_ST_DESC	Description of the alarm's logical state.	string
LOGICAL_ST_ID	ID indicating the alarm's logical state: 1=set 2=reset 3=tamper	int
PHYS_ST_ID	ID of the alarm's physical state: 1=short 2=cut 3=open 4=closed	int
PRIOR_LVL	Priority level of the alarm.	int

Table 11. Alarm Refresh Event Array (continued)

Field Name	Field Description	Data Type
PROC_ST_ID	ID of the alarm's process state. 0=active (not accurate if existing alarm)	int
REMOVE_SET	Flag that indicates if the alarm can be removed while in a set condition.	boolean
RESET_COUNT	Number of times this alarm has been reset.	int
RESPONSE_LIST	Responses for this alarm.	array
SET_COUNT	Number of times this alarm has been set.	int
TABLE_ID	Table ID of the alarm location.	int
TIME	Time of the alarm. Format in HHMMSS.	int
TIMESTAMP	Event timestamp formatted in number of seconds since Jan 1 1970.	int
TZ	Time zone of the Picture Perfect system.	string
UTC_DATE	Date of the alarm in UTC time. Format in YYYYMMDD.	int
UTC_TIME	Time of the alarm in UTC time. Format in HHMMSS.	int

Table 12. Alarm Refresh Response Array

Field Name	Field Description	Data Type
DATE	Date of alarm processing. Formatted in YYYYMMDD.	int
HOST_DATE	Date of the alarm in HOST DATE. Formatted in YYYYMMDD.	int
HOST_TIME	Time of the alarm in HOST TIME. Formatted in HHMMSS.	int
RESPONSE	Operator response for process state change.	string
TIME	Time of alarm processing. Formatted in HHMMSS.	int
TIMESTAMP	Event timestamp formatted in number of seconds since Jan 1 1970.	int
TZ	Time zone of the Picture Perfect system.	string
USER_ID	Login name of operator processing alarm.	string
UTC_DATE	Date of the alarm in UTC time. Formatted in YYYYMMDD.	int
UTC_TIME	Time of the alarm in UTC time. Formatted in HHMMSS.	int

ALarm Refresh Event Array Example (XML-RPC)

```
<methodCall>
 <methodName>ppmerlin.processAlarmRefresh</methodName>
 <params>
  <param><value><array><data><value><struct>
   <member><name>ALARM_ID</name><value><int>16</int></value></member>
   <member><name>ALARM_DESC</name><value><string>DOOR FORCED OPEN</string> </value></member>
   <member><name>DATE</name><value><int>20030405</int></value></member>
   <member><name>HOST_DATE</name><value><int>20030405</int></value></member>
   <member><name>UTC_DATE</name><value><int>20030405</int></value></member>
   <member><name>TIME</name><value><int>135827</int></value></member>
   <member><name>HOST_TIME</name><value><int>135827</int></value></member>
   <member><name>UTC_TIME</name><value><int>135827</int></value></member>
   <member><name>TZ</name><value><string>EDT</string></value></member>
   <member><name>TABLE_ID</name><value><int>16</int></value></member>
   <member><name>PRIOR_LVL</name><value><int>10</int></value></member>
   <member><name>HOST_ID</name><value><int>1</int></value></member>
   <member><name>HOST_DESC</name><value><string>ppmerlin</string></value></member>
   <member><name>LOC_ID</name><value><int>1793</int></value></member>
   <member><name>LOC_DESC</name><value><string>160-1-0 DOOR</string></value></member>
   <member><name>INGRP_ID</name><value><int>27</int></value></member>
   <member><name>LOGICAL_ST_DESC</name><value><string>STATE</string></value></member>
   <member><name>LOGICAL_ST_ID</name><value><int>2</int></value></member>
   <member><name>PHYS_ST_ID</name><value><int>4</int></value></member>
   <member><name>PROC_ST_ID</name><value><int>4</int></value></member>
   <member><name>FACILITY_ID</name><value><int>1</int></value></member>
   <member><name>FACILITY_DESC</name><value><string>FACILTY_DESC</string></value></member>
   <member><name>REMOVE_SET</name><value><boolean>0</boolean></value></member>
   <member><name>TIMESTAMP</name><value><int>1049569107</int></value></member>
   <member><name>SET_COUNT</name><value><int>3</int></value></member>
   <member><name>RESET_COUNT</name><value><int>3</int></value></member>
   <member><name>RESPONSE_LIST</name>
    <value><array><data><value><struct>
   <member><name>RESPONSE</name><value><string>LOCAL POLICE CALLED</string></value></member>
   <member><name>USER_ID</name><value><string>jsnyder</string></value></member>
   <member><name>DATE</name><value><int>20030405</int></value></member>
   <member><name>HOST_DATE</name><value><int>20030405</int></value></member>
   <member><name>UTC_DATE</name><value><int>20030405</int></value></member>
   <member><name>TIME</name><value><int>140107</int></value></member>
   <member><name>HOST_TIME</name><value><int>140107</int></value></member>
   <member><name>UTC_TIME</name><value><int>140107</int></value></member>
   <member><name>TZ</name><value><string>EDT</string></value></member>
   <member><name>TIMESTAMP</name><value><int>1049569267</int></value></member>
    </struct></value></data></array></value>
   </member>
  </struct></value></data></array></value></param>
 </params>
</methodCall>
```

Badge Event (hostname.processBadge)

The Badge Event Object contains all of the information relevant to a badge transaction that has occurred on the Picture Perfect system.

Table 13. Badge Event

Field Name	Field Description	Data Type
ACC_RGHT_DESC	Description of the badge owner's access right (category). (not supported)	string
ACC_RGHT_ID	ID of the badge owner's access rights (category).	int
ACTIVITY_TYPE	Identical to CLASSIFICATION_TYPE.	int
AREA_DESC	Ddescription of the area in which the badge was read.	string
BID	Unique ID of the badge read.	string
CLASSIFICATION_DESC	Description of the type of badge read.	string
CLASSIFICATION_TYPE	Type of badge read: 0=VALID_NORMAL=0 1=READER_UNKNOWN 2=READER_OFFLINE 3=UNKNOWN_READERTYPE 4=AREA_UNKNOWN 5=AREA_OFFLINE 6=NO_CATEGORY_MATCH 7=INVALID_APB_IN 8=INVALID_APB_OUT 9=INVALID_PIN 10=BADGE_UNKNOWN 11=BADGE_LOST 12=BADGE_SUSPENDED 13=BADGE_DELETED 14=LEARN_TIMEOUT 15=INVALID_TA_IN 16=INVALID_TA_OUT 17=INVALID_SHUNT 18=BADGE_EXPIRED 19=VALID_APB_IN 20=VALID_APB_OUT 21=VALID_TA_IN 22=VALID_TA_OUT	int

Table 13. Badge Event (continued)

Field Name	Field Description	Data Type
	23=PASSIVE_APB_IN 24=PASSIVE_APB_OUT 25=INVALID_FUNC_CODE 26=KP_INVALID_BDG 27=KP_INVALID_OPEN_DOOR 28=KP_DISABLED 29=INCOMPLETE_VALIDATE 30=DEGRADED_OPEN 31=USAGE_EXHAUSTED 32=VALID_NO_PASSAGE 33=DBL_BDG_DOOR_LOCKED 34=VALID_TOGGLE 35=VALID_NOT_UNLOCK 36=TOUR_BDG 37=VALID_NO_DOOR 38=SWIPE_AND_SHOW 39=VALID_FLOOR 40=INVALID_FLOOR 42=VALID_TIMED_APB_IN 43=INVALID_TIMED_APB_IN 44=VALID_TIMED_APB_OUT 45=INVALID_TIMED_APB_OUT	
CREATE_TIME	(not supported)	int
DATE	Date the badge was read. Format in YYYYMMDD.	int
EMP_NUM	Employee number of the badge owner.	string
FACILITY_DESC	Description of the facility where the badge was read (not supported).	string
FACILITY_ID	ID of the facility where the badge read occurred.	int
FNAME	First name of the badge owner.	string
HOST_DATE	Date the badge was read in HOST TIME. Format in YYYYMMDD.	int
HOST_DESC	Description of host that processed the badge read (not supported).	string
HOST_ID	ID of the host that processed the badge read.	int
HOST_TIME	Time the badge was read in HOST TIME. Format in HHMMSS.	int
INITS	Initials of the badge owner.	string
LNAME	Last name of the badge owner.	string
LOC_DESC	Location description of the device that read the badge.	string

Table 13. Badge Event (continued)

Field Name	Field Description	Data Type
LOC_ID	Location ID of the device that read the badge.	int
ORIG_DESC	Origination description of the device reading the badge (not supported).	string
ORIG_ID	Origination ID of device reading the badge.	int
PID	ID of the person to whom the badge belongs. (same as bid in Picture)	string
SWIPE_SHOW_ACTION	Swipe and show action that has taken place: 0=NO_ACTION (No action for swipe and show) 1=PHOTO_DOOR (photo with door control) 2=PHOTO_NO_DOOR (photo without door control)	int
TIME	Time the badge was read. Format in HHMMSS.	int
TIMESTAMP	Event timestamp formatted in number of seconds since Jan 1 1970.	int
TZ	Time zone of the Picture Perfect system.	string
UTC_DATE	Date the badge was read in UTC time. Format in YYYYMMDD.	int
UTC_TIME	Time the badge was read in UTC time. Format in HHMMSS.	int

Badge Event Example (XML-RPC)

```
<methodCall>
 <methodName>ppmerlin.processBadge</methodName>
 <params>
  <param><value><struct>
   <member><name>ORIG_ID</name><value><int>169</int></value></member>
   <member><name>ORIG_DESC</name><value><string>ORIG_DESC</string></value></member>
   <member><name>DATE</name><value><int>20030405</int></value></member>
   <member><name>HOST_DATE</name><value><int>20030405</int></value></member>
   <member><name>UTC_DATE</name><value><int>20030405</int></value></member>
   <member><name>TIME</name><value><int>131033</int></value></member>
   <member><name>HOST_TIME</name><value><int>131033</int></value></member>
   <member><name>UTC_TIME</name><value><int>131033</int></value></member>
   <member><name>TZ</name><value><string>EDT</string></value></member>
   <member><name>ACTIVITY_TYPE</name><value><int>0</int></value></member>
   <member><name>CLASSIFICATION_TYPE</name><value><int>0</int></value></member>
   <member><name>CLASSIFICATION_DESC</name><value><string>OPEN</string></value></member>
   <member><name>HOST_ID</name><value><int>1</int></value></member>
   <member><name>HOST_DESC</name><value><string>HOST-DESC</string></value></member>
   <member><name>LOC_ID</name><value><int>897</int></value></member>
   <member><name>LOC_DESC</name><value><string>160-1-0 DOOR</string></value></member>
   <member><name>AREA_DESC</name><value><string>AREA_DESC</string></value></member>
   <member><name>BID</name><value><string>152000125454</string></value></member>
   <member><name>ACC_RGHT_ID</name><value><int>-1</int></value></member>
   <member><name>ACC_RGHT_DESC</name><value><string>CATEGORY_DESC</string></value></member>
   <member><name>PID</name><value><string>152000125454</string></value></member>
   <member><name>FNAME</name><value><string>Duke</string></value></member>
   <member><name>LNAME</name><value><string>Nukem</string></value></member>
   <member><name>INITS</name><value><string>DN</string></value></member>
   <member><name>EMP_NUM</name><value><string>1234</string></value></member>
   <member><name>FACILITY_ID</name><value><int>0</int></value></member>
   <member><name>FACILITY_DESC</name><value><string>FACILITY DESC</string></value></member>
   <member><name>TIMESTAMP</name><value><int>1049566233</int></value></member>
   <member><name>CREATE_TIME</name><value><int>1877507792</int></value></member>
   <member><name>SWIPE_SHOW_ACTION</name><value><int>0</int></value></member>
  </struct></value></param>
 </params>
</methodCall>
```

Input Event (hostname.processInputActivity)

The Input Event Object contains information concerning an input transaction.

Table 14. Input Event

Field Name	Field Description	Data Type
ACTIVITY_TYPE	Type of input transaction. 0=INPUT	int
CREATE_TIME	Creation time of the input event.	int
DATE	Date of the input transaction. Format in YYYYMMDD.	int
FACILITY_ID	Facility of the input transaction.	int
HOST_DATE	Date of the input transaction in HOST TIME. Format in YYYYMMDD.	int
HOST_TIME	Time of the input transaction in HOST TIME. Format in HHMMSS.	int
IN_STATE	Input state of the input transaction. IN_STATE & 0x03 == 0x01=SHORT IN_STATE & 0x03 == 0x02=GROUND IN_STATE & 0x03 == 0x03=ERROR IN_STATE & 0x04 == 0x04=OPEN IN_STATE & 0x04 !== 0x04=CLOSED	int
RECORD_DESCRIPTION	Description of the input.	string
TIME	Time of the input transaction. Format in HHMMSS.	int
TIMESTAMP	Event timestamp formatted in number of seconds since Jan 1 1970.	int
TZ	Time zone that correlates to the timestamp.	string
UTC_DATE	Date of the input transaction in UTC time. Format in YYYYMMDD.	int
UTC_TIME	Time of the input transaction in UTC time. Format in HHMMSS.	int

Operator Event (hostname.processOperActivity)

The Operator Event Object contains information concerning an operator transaction.

Table 15. Operator Event

Field Name	Field Description	Data Type
DATE	Date of the operator transaction in OPERATOR TIME. Format in YYYYMMDD.	int
EMP_NUM	Employee number of the operator that generated this transaction.	string
FIELD_ID	Field ID (description) of record affected by the operator.	string
HOST_DATE	Date of the operator transaction in HOST TIME. Format in YYYYMMDD.	int
HOST_TIME	Time of the operator transaction in HOST TIME. Format in HHMMSS.	int

Table 15. Operator Event (continued)

Field Name	Field Description	Data Type
LOGIN_ID	Login ID of the operator that generated this transaction.	string
RECORD_DESCRIPTION	Description of the record affected by the operator.	string
TABLE_ID	Table ID of the record affected by the operator.	string
TIME	Time of the operator transaction in OPERATOR TIME. Format in HHMMSS	int
TIMESTAMP	Event timestamp formatted in number of seconds since Jan 1 1970.	int
TRANSACTION _TYPE	Type of operator transaction: 1=LOGIN 2=LOGOUT 3=UPDATE 4=DELETE 5=QUERY 6=COMMAND 7=STATUS 8=SHUTDOWN 9=INSERT 10=UNKNOWN	int
TZ	Time zone that correlates to the timestamp.	string
UTC_DATE	Date of the operator transaction in UTC time. Format in YYYYMMDD.	int
UTC_TIME	Time of the operator transaction in UTC time. Format in HHMMSS.	int
VALUE	Value of the change to the record affected by the operator.	string

Popup Event (hostname.processPopup)

The Popup Event Object contains information concerning an informational popup that displays to an operator with “Receive System Notifications” enabled.

Table 16. Popup Event

Field Name	Field Description	Data Type
MESSAGE_CONTENT	Content of a popup message.	string
MESSAGE_TYPE	Type of popup message: 0=WARNING 1=ERROR 2=INFO 3=MESSAGE 4=QUESTION 5=WORKING	int

PPRS Mode Change Event (hostname.processPPRSModeChange)

The PPRS Mode Change Event Object contains the new state of the Picture Perfect server, either Primary or Backup. It is generated when the server changes its state.

Table 17. PPRS Mode Change Event

Field Name	Field Description	Data Type
PPRS_MODE	New state of the Picture Perfect host: 1=PRIMARY 2=BACKUP	int

Single Alarm Refresh Event (hostname.processSingleAlarmRefresh)

The Single Alarm Refresh Event Object contains information concerning the state of a single Picture Perfect alarm buffer entry. This message is transmitted to an operator when this alarm becomes routed to an operator when it previously was not being routed.

Table 18. Single Alarm Refresh Event

Field Name	Field Description	Data Type
ALARM_DESC	Description of the alarm.	string
ALARM_ID	ID of the alarm.	int
DATE	Date of the alarm. Format in YYYYMMDD.	int
FACILITY_DESC	Description of the facility of the alarm location. (not supported)	string
FACILITY_ID	ID of the facility of the alarm location.	int
HOST_DATE	Date of the alarm in HOST TIME. Format in YYYYMMDD.	int
HOST_DESC	Description of host that processed the alarm. (not supported)	string
HOST_ID	ID of the host that processed the alarm.	int
HOST_TIME	Time of the alarm in the HOST TIME format in HHMMSS.	int
INGRP_ID	ID of the alarm's input group.	int
LOC_DESC	Location description of the alarm point.	string
LOC_ID	Location ID of the alarm point.	int
LOGICAL_ST_DESC	Description of the alarm's logical state.	string
LOGICAL_ST_ID	ID indicating the alarm's logical state: 1=set 2=reset 3=tamper	int

Table 18. Single Alarm Refresh Event (continued)

Field Name	Field Description	Data Type
PHYS_ST_ID	ID of the alarm's physical state: 1=short 2-cut 3=open 4=closed	int
PRIOR_LVL	Priority level of the alarm.	int
PROC_ST_ID	ID of the alarm's process state: 0=active 1=bumped 2=notified 3=remote 4=pending 5=completed 6=removed 7=purged	int
REMOVE_SET	Flag that indicates if the alarm can be removed.	boolean
REQUESTOR_ID	Unique identifier that was issued in the force alarm refresh request that is tagged to this event so that the appropriate entity receives this event.	int
RESET_COUNT	Count of the number of times this alarm has been reset in the Picture Perfect alarm buffer.	int
SET_COUNT	Count of the number of times this alarm has been set in the Picture Perfect alarm buffer.	int
TABLE_ID	Table ID of the alarm location.	int
TIME	Time of the alarm. Format in HHMMSS.	int
TIMESTAMP	Event timestamp formatted in number of seconds since Jan 1 1970.	int
TZ	Time zone of the Picture Perfect system.	string
UTC_DATE	Date of the alarm in UTC time. Format in YYYYMMDD.	int
UTC_TIME	Time of the alarm in UTC time. Format in HHMMSS.	int

Table 19. Single Alarm Refresh Response Array

Field Name	Field Description	Data Type
DATE	Date of alarm processing. Formatted in YYYYMMDD.	int
HOST_DATE	Date of alarm processing in HOST TIME. Formatted in YYYYMMDD.	int
HOST_TIME	Time of alarm processing in HOST TIME. Formatted in HHMMSS.	int

Table 19. Single Alarm Refresh Response Array

Field Name	Field Description	Data Type
RESPONSE	Operator response for process state change.	string
TIME	Time of alarm processing. Formatted in HHMMSS.	int
TIMESTAMP	Event timestamp formatted in number of seconds since Jan 1 1970.	int
TZ	Time zone of the Picture Perfect system.	string
USER_ID	Login name of operator processing alarm.	string
UTC_DATE	Date of the alarm processing in UTC time. Formatted in YYYYMMDD.	int
UTC_TIME	Time of the alarm processing in UTC time. Formatted in HHMMSS.	int

Single Alarm Refresh Event Example (XML-RPC)

```
<methodCall>
 <methodName>ppmerlin.ProcessSingleAlarmRefresh</methodName>
 <params>
  <param><value><struct>
   <member><name>ALARM_ID</name><value><int>16</int></value></member>
   <member><name>ALARM_DESC</name><value><string>DOOR FORCED OPEN</string> </value></member>
   <member><name>DATE</name><value><int>20030405</int></value></member>
   <member><name>HOST_DATE</name><value><int>20030405</int></value></member>
   <member><name>UTC_DATE</name><value><int>20030405</int></value></member>
   <member><name>TIME</name><value><int>135827</int></value></member>
   <member><name>HOST_TIME</name><value><int>135827</int></value></member>
   <member><name>UTC_TIME</name><value><int>135827</int></value></member>
   <member><name>TZ</name><value><string>EDT</string></value></member>
   <member><name>TABLE_ID</name><value><int>16</int></value></member>
   <member><name>PRIOR_LVL</name><value><int>10</int></value></member>
   <member><name>HOST_ID</name><value><int>1</int></value></member>
   <member><name>HOST_DESC</name><value><string>ppmerlin</string></value></member>
   <member><name>LOC_ID</name><value><int>1793</int></value></member>
   <member><name>LOC_DESC</name><value><string>160-1-0 DOOR</string></value></member>
   <member><name>INGRP_ID</name><value><int>27</int></value></member>
   <member><name>LOGICAL_ST_DESC</name><value><string>STATE</string></value></member>
   <member><name>LOGICAL_ST_ID</name><value><int>2</int></value></member>
   <member><name>PHYS_ST_ID</name><value><int>4</int></value></member>
   <member><name>PROC_ST_ID</name><value><int>4</int></value></member>
   <member><name>FACILITY_ID</name><value><int>1</int></value></member>
   <member><name>FACILITY_DESC</name><value><string>FACILTY_DESC</string></value></member>
   <member><name>REMOVE_SET</name><value><boolean>0</boolean></value></member>
   <member><name>REQUESTOR_ID</name><value><int>1</int></value></member>
   <member><name>TIMESTAMP</name><value><int>1049569107</int></value></member>
   <member><name>SET_COUNT</name><value><int>3</int></value></member>
   <member><name>RESET_COUNT</name><value><int>3</int></value></member>
   <member><name>RESPONSE_LIST</name>
    <value><array><data><value><struct>
   <member><name>RESPONSE</name><value><string>LOCAL POLICE CALLED</string></value></member>
   <member><name>USER_ID</name><value><string>jsnyder</string></value></member>
   <member><name>DATE</name><value><int>20030405</int></value></member>
   <member><name>HOST_DATE</name><value><int>20030405</int></value></member>
   <member><name>UTC_DATE</name><value><int>20030405</int></value></member>
   <member><name>TIME</name><value><int>140107</int></value></member>
   <member><name>HOST_TIME</name><value><int>140107</int></value></member>
   <member><name>UTC_TIME</name><value><int>140107</int></value></member>
   <member><name>TZ</name><value><string>EDT</string></value></member>
   <member><name>TIMESTAMP</name><value><int>1049569267</int></value></member>
    </struct></value></data></array></value>
   </member>
  </struct></value></param>
 </params>
</methodCall>
```

Tour Event (hostname.processTourActivity)

The Tour Event Object contains information concerning a tour transaction.

Table 20. Tour Event

Field Name	Field Description	Data Type
AREA_DESC	Description of the area.	string
CREATE_TIME	Creation time of the tour event.	int
DATE	Date of the tour transaction. Format in YYYYMMDD.	int
EMP_NUM	Employee number of the individual performing the tour.	string
FACILITY_ID	Facility of the tour transaction.	int
HOST_DATE	Date of the tour transaction in HOST TIME. Format in YYYYMMDD.	int
HOST_TIME	Time of the tour transaction in HOST TIME. Format in HHMMSS.	int
IN_STATE	Input state of the tour transaction. IN_STATE & 0x03 == 0x01=SHORT IN_STATE & 0x03 == 0x02=GROUND IN_STATE & 0x03 == 0x03=ERROR IN_STATE & 0x04 == 0x04=OPEN IN_STATE & 0x04 !== 0x04=CLOSED	int
INITS	Initials of the individual performing the tour.	string
LNAME	Last name of the individual performing the tour.	string
TIME	Time of the tour transaction. Format in HHMMSS.	int
TOUR_DESC	Description of the tour.	string
TOUR_POINT_DESC	Description of the tour point.	string
TOUR_STATUS	Status of the tour.	int
TOUR_SUSPEND_TIME	Suspend duration of the tour.	int
TOUR_TYPE	Type of tour transaction: 5=Digital Input 6=Badge Input 7=Automatically End 8=Start 9=Stop 10=Suspended 11=Resumed 12=Automatically Resumed	int
UTC_DATE	Date of the tour transaction in UTC time. Format in YYYYMMDD.	int
UTC_TIME	Time of the tour transaction in UTC time. Format in HHMMSS.	int

Event Response Object

All events sent from the eifevtmgr expect a response object containing a boolean result.

Table 21. Event Response

Field Name	Field Description	Data Type
RESULT	Success flag of event call.	boolean

Event Response Example Message

```
<methodResponse>
 <params>
  <param><value><struct>
   <member><name>RESULT</name><value><boolean>1</boolean></value></member>
  </struct></value></param>
 </params>
</methodResponse>
```

EIF Commands

For each of the commands that call into Picture Perfect, the session is treated as one parameter and the rest of the fields are treated as an object. All results are returned either as an object or an array of objects.

This section describes the Commands supported by the API. The name of the Command and all of its input and output fields are listed.

Table 22. EIF Commands Method

Command Name	Method Name	Command Description
Activate Output	picture.activateOutput	Activates an output.
Force Alarm Refresh	picture.alarmRefresh	Forces a refresh of all alarms in the Picture Perfect alarm buffer.
Change Mode	picture.changeMode	Changes the mode for all micros in the given facility.
Check or Add Port	picture.checkAddPort	Checks the existence of the specified port and adds it to the **`/etc/services`** if it does not already exist.
Disable Input Group on Micro	picture.disableIngrp	Disables an input group.
Disable Input	picture.disableInput	Disables an input.
Enable Input Group on Micro	picture.enableIngrp	Enables an input group.
Enable Input	picture.enableInput	Enables an input.
Logout	picture.logout	Terminates a login session by making a session ID invalid.
Lock Door	picture.lockDoor	Locks a door.
Create Operator History	picture.operatorHistory	Generates an operator history transaction.

Table 22. EIF Commands Method (continued)

Command Name	Method Name	Command Description
Change Mode of Multiple Facilities.	picture.ppChangeMode	Changes the mode for all micros in the given facilities.
Validate Printer Queue	picture.printerqueue	Validates the specified printer queue.
Reset Micro	picture.resetMicro	Resets the specified micro.
Reset Output	picture.resetOutput	Resets an output.
Reset Timed APB	picture.resetTimedAPB	Resets the Timed APB status of the specified badges.
Set Diagnostics	picture.setDiagnostics	Enables or disables the specified diagnostics.
Set Ping Interval	picture.setPingInterval	Sets the ping interval.
Tour Functions	picture.tourFunctions	Performs the specified Guard Tours function.
Validate TTY	picture.ttyValidate	Validates the specified tty port.
Unlock Door	picture.unlockDoor	Unlocks a door.
Process Alarm	picture.updateAlarmProcessingState	Responds to an alarm in the Picture Perfect alarm buffer.

Command Protocol

Picture Perfect 4.0 EIF Commands always return a boolean success value, and nothing more. The success value explains whether or not the command was executed. This differs from EIF Requests in that a request will return a boolean success value along with additional information.

Activate Output (picture.activateOutput)

This command activates the specified output for the specified activation time.

Table 23. Activate Output Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
OUTPUT_ID	ID of the output being activated.	int
ACTIVATION_TIME	Amount of time for which to activate the output: 0=default -1=indefinite >0=number of seconds	int

Table 24. Activate Output Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Activate Output Example Message

```
<methodCall>
 <methodName>picture.activateOutput</methodName>
 <params>
  <param><value>RU65IRPSWZ</value></param>
  <param><value><struct>
   <member><name>OUTPUT_ID</name><value><int>897</int></value></member>
   <member><name>ACTIVATION_TIME</name><value><int>0</int></value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct><member><name>RESULT</name><value><boolean>1</boolean></value></member> </struct></value>
  </param>
 </params>
</methodResponse>
```

Force Alarm Refresh (picture.alarmRefresh)

This command forces a refresh of all alarms in the Picture Perfect buffer to be sent to the client by a subsequent **`hostname.processAlarmRefresh`** EIF Event.

Table 25. Force Alarm Refresh Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 26. Force Alarm Refresh Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Force Alarm Refresh Example Message

```
<methodCall>
 <methodName>picture.alarmRefresh</methodName>
 <params>
  <param><value>I65DTVAWED</value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
     <member><name>RESULT</name><value><boolean>1</boolean></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

Change Mode (picture.changeMode)

This command changes the mode for all micros in the specified facility to the specified mode.

Table 27. Change Mode Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
FACILITY_ID	ID of the facility for which all micros should have their mode changed.	int
MODE_ID	ID of the mode to change to.	int
OVERRIDE	Whether or not to set the TP_OVERRIDE_AUTO_MODE bit in the UCM message to be sent to each micro: 1=Set the TP_OVERRIDE_AUTO_MODE bit 2=Do not	boolean

Table 28. Change Mode Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfull.y	boolean

Change Mode Example Message

```
<methodCall>
 <methodName>picture.changeMode</methodName>
 <params>
  <param><value>L8DN087B2C</value></param>
    <param><value><struct>
     <member><name>FACILITY_ID</name><value><int>-1</int></value></member>
     <member><name>MODE_ID</name><value><int>5</int></value></member>
     <member><name>OVERRIDE</name><value><int>0</int></value></member>
   </struct></value></param>
 </params>
<methodCall>

</methodResponse>
  <params>
   <param><
    <value><struct>
     <member><name>RESULT</name><value><boolean>1</boolean></value></member>
    </struct></value>
   </param>
 </params>
<methodResponse>
```

Check or Add Port (picture.checkAddPort)

This command checks for the existence of the specified port and adds it to the **`/etc/services`** file if it does not already exist.

Table 29. Change Mode Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
N/A	Port to be checked or added.	string
N/A	Hostname upon which the port should be checked or added.	string

Table 30. Change Mode Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the port already existed, False if the port was added.	boolean

Check or Add Port Example Message

```
<methodCall>
 <methodName>picture.checkAddPort</methodName>
 <params>
  <param><value>L8DN087B2C</value></param>
  <param><value>6202</value></param>
  <param><value>picture</value></param>
 </params>
<methodCall>

</methodResponse>
  <params>
   <param><
    <value><struct>
     <member><name>RESULT</name><value><boolean>1</boolean></value></member>
    </struct></value>
   </param>
 </params>
<methodResponse>
```

Disable Input Group on Micro (picture.disableIngrp)

This command disables the specified input group.

Table 31. Disable Input Group on Micro Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
INGRP_ID	ID of the input group being disabled.	int

Table 32. Disable Input Group on Micro Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Disable Input Group on Micro Example Message

```
<methodCall>
 <methodName>picture.disableIngrp</methodName>
 <params>
  <param><value>RU65IRPSWZ</value></param>
  <param><value><struct>
   <member><name>INGRP_ID</name><value><int>234</int></value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct><member><name>RESULT</name><value><boolean>1</boolean></value></member> </struct></value>
  </param>
 </params>
</methodResponse>
```

Disable Input (picture.disableInput)

This command disables the specified input.

Table 33. Disable Input Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
INPUT_ID	ID of the input being disabled.	int

Table 34. Disable Input Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Disable Input Example Message

```
<methodCall>
 <methodName>picture.disableInput</methodName>
 <params>
  <param><value>RU65IRPSWZ</value></param>
  <param><value><struct>
   <member><name>INPUT_ID</name><value><int>1796</int></value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct><member><name>RESULT</name><value><boolean>1</boolean></value></member> </struct></value>
  </param>
 </params>
</methodResponse>
```

Enable Input Group (picture.enableIngrp)

This command enables the specified input group.

Table 35. Enable Input Group on Micro Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
INGRP_ID	ID of the input group being enabled.	int

Table 36. Enable Input Group on Micro Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Enable Input Group Example Message

```
<methodCall>
 <methodName>picture.enableIngrp</methodName>
 <params>
  <param><value>RU65IRPSWZ</value></param>
  <param><value><struct>
   <member><name>INGRP_ID</name><value><int>234</int></value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct><member><name>RESULT</name><value><boolean>1</boolean></value></member> </struct></value>
  </param>
 </params>
</methodResponse>
```

Enable Input (picture.enableInput)

This command enables the specified input.

Table 37. Enable Input Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
INPUT_ID	ID of the input being enabled.	int

Table 38. Enable Input Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Enable Input Example Message

```
<methodCall>
 <methodName>picture.enableInput</methodName>
 <params>
  <param><value>RU65IRPSWZ</value></param>
  <param><value><struct>
   <member><name>INPUT_ID</name><value><int>1796</int></value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct><member><name>RESULT</name><value><boolean>1</boolean></value></member> </struct></value>
  </param>
 </params>
</methodResponse>
```

Lock Door (picture.lockDoor)

This command locks the specified door.

Table 39. Lock Door Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
DOOR_ID	Door being locked.	int

Table 40. Lock Door Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Lock Door Example Message

```
<methodCall>
 <methodName>picture.lockDoor</methodName>
 <params>
  <param><value>RU65IRPSWZ</value></param>
  <param><value><struct>
   <member><name>DOOR_ID</name><value><int>897</int></value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct><member><name>RESULT</name><value><boolean>1</boolean></value></member> </struct></value>
  </param>
 </params>
</methodResponse>
```

Logout (picture.logout)

This command terminates the specified SESSION ID.

Table 41. Logout Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
UTC_TIME	(Used for logging operator history) Date and Time in UTC TIME. Format in YYYYMMDDHHMMSS. Ex: 20041215113000 = December 15th, 2004 11:30AM	string
OPERATOR_TIME	(Used for logging operator history) Date and Time in OPERATOR TIME. Format in YYYYMMDDHHMMSS. Ex: 20041215113000 = December 15th, 2004 11:30AM	string

Table 42. Logout Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the logout was successful.	boolean

Logout Example Message

```
<methodCall>
 <methodName>picture.logout</methodName>
 <params>
  <param>
   <value>BKTT9R32XU</value>
  <param>
  <param><value><struct>
   <member><name>UTC_TIME</name><value><string>20041215113000</string></value></member>
   <member><name>OPERATOR_TIME</name><value><string>20041215113000</string></value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct><member><name>RESULT</name><value><boolean>1</boolean></value></member> </struct></value>
  </param>
 </params>
</methodResponse>
```

Create Operator History (picture.operatorHistory)

This command generates the specified operator history transaction, writes it to operator history, and causes the appropriate **`hostname.processOperActivity`** EIF Events to be transmitted to all active sessions.

Table 43. Create Operator History Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
UTC_TIME	Date and Time in UTC TIME. Format in YYYYMMDDHHMMSS. Ex: 20041215113000 = December 15th, 2004 11:30AM	string
OPERATOR_TIME	Date and Time in OPERATOR TIME. Format in YYYYMMDDHHMMSS. Ex: 20041215113000 = December 15th, 2004 11:30AM	string
OPERATOR_ID	ID of the operator that is generating this transaction.	int
EMPLOYEE_ID	ID of the employee that is generating this transaction.	string
LOGIN_ID	Login ID of the operator that is generating this transaction.	string
TERMINAL_ID	ID of the terminal to which this operator transaction is associated.	int
TRANSACTION_TYPE	Type of operator transaction: 1=LOGIN 2=LOGOUT 3=UPDATE 4=DELETE 5=QUERY 6=COMMAND 7=STATUS 8=SHUTDOWN 9=INSERT 10=UNKNOWN	int
TABLE_ID	Table ID of the record affected by the operator.	string
RECORD_DESCRIPTION	Description of the record affected by the operator.	string
FACILITY_ID	ID of the facility to which this operator transaction will belong.	string
N/A	The name of the field that has been changed as a string immediately followed by the value it was changed to as a string. This pair of data can be repeated for field/value pairs. The end of the list is denoted by a field name of “MASTER EOF”.	string, string

Table 44. Create Operator History Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was successful.	boolean

Create Operator History Example Message

```
<methodCall>
 <methodName>picture.operatorHistory</methodName>
 <params>
  <param><value>L8DN087B2C</value><param>
  <param><value><struct>
   <member><name>UTC_TIME</name><value><string>20041215113000</string></value></member>
   <member><name>OPERATOR_TIME</name><value><string>20041215113000</string></value></member>
   <member><name>OPERATOR_ID</name><value><int>1</int></value></member>
   <member><name>EMPLOYEE_ID</name><value><string>E30</string></value></member>
   <member><name>LOGIN_ID</name><value><string>install</string></value></member>
   <member><name>TERMINAL_ID</name><value><int>-3</int></value></member>
   <member><name>TRANSACTION_TYPE</name><value><int>3</int></value></member>
   <member><name>TABLE_ID</name><value><string>43</string></value></member>
   <member><name>RECORD_DESCRIPTION</name><value><string>1-1-0 LOBBY DOOR</string></value></member>
   <member><name>FACILITY_ID</name><value><string>-1</string></value></member>
  </struct></value></param>
  <param><value>description</value><param>
  <param><value>1-1-0 WEST LOBBY DOOR</value><param>
  <param><value>facility</value><param>
  <param><value>-1</value><param>
  <param><value>MASTER EOF</value><param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
    <member><name>RESULT</name><value><boolean>1</boolean></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

Change Modes (picture.ppChangeMode)

This command changes the mode for all micros in the specified facilities and logs appropriate operator history.

Table 45. Change Modes Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
MODE_ID	ID of the mode to change to.	int
OVERRIDE	Whether or not to set the TP_OVERRIDE_AUTO_MODE bit in the UCM message to be sent to each micro. 1=Set the TP_OVERRIDE_AUTO_MODE bit 2=Do not	boolean
UTC_TIME	Date and Time in UTC TIME. Format in YYYYMMDDHHMMSS. Ex: 20041215113000 = December 15th, 2004 11:30AM	string
OPERATOR_TIME	Date and Time in OPERATOR TIME. Format in YYYYMMDDHHMMSS Ex: 20041215113000 = December 15th, 2004 11:30AM	string
OPERATOR_ID	ID of the operator that is changing the mode.	int
EMPLOYEE_ID	Employee ID of the operator that is changing the mode.	string
LOGIN_ID	Login ID of the operator that is generating this transaction.	string
N/A	Individual facility Ids whose modes should be changed. See "Change Modes Example Message" on page 1.41. The end of the llist is denoted by a fieldname of "MASTER EOF."	int

Table 46. Change Modes Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Change Modes Example Message

```
<methodCall>
 <methodName>picture.ppChangeMode</methodName>
 <params>
  <param><value>L8DN087B2C</value><param>
  <param><value><struct>
   <member><name>MODE_ID</name><value><int>5</int></value></member>
   <member><name>OVERRIDE</name><value><int>0</int></value></member>
   <member><name>UTC_TIME</name><value><string>200412151130</string></value></member>
   <member><name>OPERATOR_TIME</name><value><string>200412151130</string></value></member>
   <member><name>OPERATOR_ID</name><value><int>1</int></value></member>
   <member><name>EMPLOYEE_ID</name><value><string>E230</string></value></member>
   <member><name>LOGIN_ID</name><value><string>install</string></value></member>
  </struct></value></param>
  <param><value>description</value><param>
  <param><value>-1</value><param>
  <param><value>1</value><param>
  <param><value>2</value><param>
  <param><value>MASTER EOF</value><param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
    <member><name>RESULT</name><value><boolean>1</boolean></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

Validate Printer Queue (picture.printerqueue)

This command validates the specified printer queue.

Table 47. Validate Printer Queue Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
N/A	Name of the printer queue to validate.	string

Table 48. Validate Printer Queue Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Validate Printer Queue Example Message

```
<methodCall>
 <methodName>picture.printerqueue</methodName>
 <params>
  <param><value>L8DN087B2C</value></param>
  <param><value>np1200</value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
    <member><name>RESULT</name><value><boolean>1</boolean></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

Reset Micro (picture.resetMicro)

This command resets the specified micro.

Table 49. Reset Micro Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
MICRO_ID	ID of the micro being reset.	int
CLEAR	(Expert Use Only) Determines whether or not all of the badge records known to this micro should be cleared from the micro_relation table. Unless told otherwise, always set this to "0". Must be one of the following: 0=nothing 1=clear badge records from micro_relation table	int

Table 50. Reset Micro Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Reset Micro Example Message

```
<methodCall>
 <methodName>picture.resetMicro</methodName>
 <params>
  <param><value>L8DN087B2C</value></param>
  <param><value><struct>
   <member><name>MICRO_ID</name><value><int>13</int></value></member>
   <member><name>CLEAR</name><value><int>0</int></value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
    <member><name>RESULT</name><value><boolean>1</boolean></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

Reset Output (picture.resetOutput)

This command resets the specified output.

Table 51. Reset Output Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
OUTPUT_ID	ID of the output being reset.	int

Table 52. Reset Output Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Reset Output Example Message

```
<methodCall>
 <methodName>picture.resetOutput</methodName>
 <params>
  <param><value>RU65IRPSWZ</value></param>
  <param><value><struct>
   <member><name>OUTPUT_ID</name><value><int>897</int></value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct><member><name>RESULT</name><value><boolean>1</boolean></value></member> </struct></value>
  </param>
 </params>
</methodResponse>
```

Reset Timed APB (picture.resetTimedAPB)

This command resets the timed APB status of the specified badges.

Table 53. Reset Timed APB Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
N/A	ID of the badge record as an integer immediately followed by the badge encode number of the badge records as a string. This pair of data can be repeated for multiple badges.	int, string

Table 54. Reset Timed APB Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Reset Timed APB Example Message

```
<methodCall>
 <methodName>picture.resetTimedAPB</methodName>
 <params>
  <param><value>L8DN087B2C</value></param>
  <param><value>5301</value></param>
  <param><value>154020397467</value></param>
  <param><value>908</value></param>
  <param><value>155273691823</value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct><member><name>RESULT</name><value><boolean>1</boolean></value></member> </struct></value>
  </param>
 </params>
</methodResponse>
```

Set Diagnostics (picture.setDiagnostics)

This command enables or disables the specific diagnostics.

Table 55. Set Diagnostics Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
DIAGNOSTICS	Long decimal value that contains all of the bitwise diagnostics flags that are to be enabled or disabled. The bitwise diagnostic flags are as follows: mumgr=1073741824 eifreqmgr=536870912 eifevtmgr=268435456 eflash=134217728 ucs=67108864 evtmgr=33554432 keymgr=16777216 cfgmgr=8388608 nwmdrv=4194304 pcsrvd=2097152 tracer=1048576 tcmgr=524288 mrtmgr=262144 tourmgr=131072 timer=65536 log=32768 schmgr=16384 stsmgr=8192 mreset=4096 rsndmgr=2048 snddrv=1024 rcvdrv=512 moddrv=256 oprmgr=128 almmgr=64 bdgmgr=32 prmgr=16 dbmgr=8 sndmgr=4 rcvmgr=2 main=1	string
ENABLED	Flag to determine whether the DIAGNOSTICS specified should be enabled or disabled.	int
LEVEL	Target diagnostics level (0-9).	int

Table 56. Set Diagnostics Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Set Diagnostics Example Message

```
<methodCall>
 <methodName>picture.setDiagnostics</methodName>
 <params>
  <param><value>L8DN087B2C</value></param>
  <param><value><struct>
   <member><name>DIAGNOSTICS</name><value><string>64</string></value></member>
   <member><name>ENABLED</name><value><int>1</int></value></member>
   <member><name>LEVEL</name><value><int>9<int></value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
    <member><name>RESULT</name><value><boolean>1</boolean></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

Set Ping Interval (picture.setPingInterval)

This command sets the specified ping interval

Table 57. Set Ping Interval Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
PING_INTERVAL	Number of seconds to set the ping interval. 0=disabled	int
VERSION	Version of the interface.	string

Table 58. Set Ping Interval Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Set Ping Interval Example Message

```
<methodCall>
 <methodName>picture.setPingInterval</methodName>
 <params>
  <param><value>L8DN087B2C</value></param>
  <param><value><struct>
   <member><name>PING_INTERVAL</name><value><int>5</int></value></member>
   <member><name>VERSION</name><value>v0.5.5</value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
    <member><name>RESULT</name><value><boolean>1</boolean></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

Tour Functions (picture.tourFunctions)

This command performs the specified Guard Tours function.

Table 59. Tour Functions Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
UTC_TIME	Epoch time in seconds UTC.	string
LOGIN_ID	Login ID of the guard performing the tour function.	string
TOUR_ID	ID of the tour for which the function is being performed.	string
TOUR_ACTION	Action being performed for the tour, which must be one of the following: START STOP SUSPEND RESUME	string
BID	Badge encode number of the badge involved in this tour.	string
TOUR_SUSPEND_TIME	Suspend time for this tour.	string
TOUR_EXCEPTION_CODE	ID of the exception code for this tour function.	string
TOUR_COMMENT	Comment for this tour function.	string

Table 60. Tour Functions Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Tour Functions Example Message

```
<methodCall>
 <methodName>picture.tourFunctions</methodName>
 <params>
  <param><value>L8DN087B2C</value></param>
  <param><value><struct>
   <member><name>UTC_TIME</name><value><string>19382712</string></value></member>
   <member><name>LOGIN_ID</name><value>string>install</string></value></member>
   <member><name>TOUR_ID</name><value>string>1</string></value></member>
   <member><name>TOUR_ACTION</name><value>string>START</string></value></member>
   <member><name>BID</name><value>string>1542736879921</string></value></member>
   <member><name>TOUR_SUSPEND_TIME</name><value>string>20</string></value></member>
   <member><name>TOUR_EXCEPTION_CODE</name><value>string>4</string></value></member>
   <member><name>TOUR_COMMENT</name><value>string>beginning it</string></value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
    <member><name>RESULT</name><value><boolean>1</boolean></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

TTY Validate (picture.ttyValidate)

This command validates the specified tty port.

Table 61. TTY Validate Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
TTY_NAME	Name of the TTY port.	string
MODEM_TYPE	Modem type (the ID of the appropriate modem record in the modem table).	int

Table 62. TTY Validate Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

TTY Validate Example Message

```
<methodCall>
 <methodName>picture.TTYvalidate</methodName>
 <params>
  <param><value>L8DN087B2C</value></param>
  <param><value>/dev/ttyD001</value></param>
  <param><value>1</param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
    <member><name>RESULT</name><value><boolean>1</boolean></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

Unlock Door (picture.unlockDoor)

This command unlocks the specified door for the specified amount of time.

Table 63. Unlock Door Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
OUTPUT_ID	Output of the door being unlocked.	int
UNLOCK_TIME	Amount of time to unlock the door for: 0=default -1=indefinite >0=number of seconds	int
DOOR_ID	Door being unlocked.	int

Table 64. Unlock Door Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully.	boolean

Unlock Door Example Message

```
<methodCall>
 <methodName>picture.unlockDoor</methodName>
 <params>
  <param><value>RU65IRPSWZ</value></param>
  <param><value><struct>
   <member><name>OUTPUT_ID</name><value><int>897</int></value></member>
   <member><name>UNLOCK_TIME</name><value><int>0</int></value></member>
   <member><name>DOOR_ID</name><value><int>897</int></value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct><member><name>RESULT</name><value><boolean>1</boolean></value></member> </struct></value>
  </param>
 </params>
</methodResponse>
```

Process Alarm (picture.updateAlarmProcessingState)

This command responds to the alarm specified by the alarm ID, input group ID, and the input ID in the Picture Perfect alarm buffer.

Table 65. Process Alarm Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
RESPONSE	Operator response.	string
USER_ID	Operator login name.	string
HOST_ID	Alarm host record ID.	int
INGRP_ID	Alarm input group ID.	int
LOC_ID	Location/Input record ID.	int
PROC_ST_ID	Processing state ID: 4=acknowledge 5=remove 7=purge	int
ALARM_ID	Alarm record ID.	int

Table 66. Process Alarm Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the command was processed successfully	boolean

Process Alarm Example Message

```
<methodCall>
 <methodName>picture.updateAlarmProcessingState</methodName>
 <params>
  <param><value>LSIP0TAPWW</value></param>
  <param><value><struct>
   <member><name>RESPONSE</name><value>AREA IS FLOODED</value></member>
   <member><name>USER_ID</name><value>admin</value></member>
   <member><name>HOST_ID</name><value><int>1</int></value></member>
   <member><name>INGRP_ID</name><value><int>17</int></value></member>
   <member><name>LOC_ID</name><value><int>200</int></value></member>
   <member><name>PROC_ST_ID</name><value><int>4</int></value></member>
   <member><name>ALARM_ID</name><value><int>7</int></value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
    <member><name>RESULT</name><value><boolean>1</boolean></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

EIF Data Requests

This section describes the data requests supported by the API.

EIF Data Requests Method Table

Table 67. EIF Data Requests Method

Command Name	Method Name	Command Description
Convert Date and Time	picture.convertDateTime	Converts the specified date/time/timezone into HOST TIME.
Directory Listing	picture.directoryListing	Lists the contents of the specified directory on the host.
Get Alarm Status	picture.getAlarmStatus	Get the general alarm status information.
Get Configured Devices	picture.getConfiguredDevices	Get the counts of configured devices on the system.
Get Diagnostics	picture.getDiagnostics	Get the diagnostics that are currently configured.
Get History Counts	picture.getHistoryCounts	Get the history counts.
Get History Flags	picture.getHistoryFlags	Get the history time to archive flags.
Get Host ID	picture.getHostId	Get the host ID.
Get License Information	picture.getLicenseInformation	Get the license information for all licensed packages.
Get Operators Logged In	picture.getOperatorsLoggedIn	Get the session details of each operator currently logged in.
Get Time	picture.getTime	Get the current date and time in the specified time zone.
Get Host Time	picture.HostTime	Get the host date and time.
Login	picture.loginEIF	Authenticates a user ID and password and creates a valid session.
Get Micro Time	picture.microTime	Get the date and time for the specified micro.
Query Access Points	picture.queryAccessPoints	Query for all access points on the host.
Query Alarm Instructions	picture.queryAlarmInstructions	Query for all alarm instructions for a particular alarm on the host.
Query Alarm Responses	picture.queryAlarmResponses	Query for all alarm responses on the host.
Query Facilities	picture.queryFacilities	Query for all facilities on the host.
Query Input Groups	picture.queryInputGroups****	Query for all input groups on the host.
Query Inputs	picture.queryInputs	Query for all inputs on the host.
Query Micro State	picture.queryMicroState	Query for the state of the specified micro.
Query Modes	picture.queryModes	Query for all modes on the host.
Query Operators	picture.queryOperators	Query for all operators on the host.
Query Outputs	picture.queryOutputs	Query for all outputs on the host.
Query Port State	picture.queryPortState	Query for the state of the specified port.
Query Status	picture.queryStatus	Query for status from micro.

Request Protocol

If an EIF request is successfully executed, Picture Perfect 4.0 returns a boolean value along with additional information in the form of an object or an array of objects.

Convert Date and Time (picture.convertDateTime)

This request converts the specified date and time in the specified time zone into HOST TIME.

Table 68. Convert Date and Time Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
N/A	Source date in YYYYMMDD format.	int
N/A	Source time in HHMMSS format.	int
N/A	Source time zone ID.	int

Table 69. Convert Date and Time Output Parameters

Field Name	Field Description	Data Type
CONVERTED_DATE	Converted date in HOST TIME. Format is YYYYMMDD.	int
CONVERTED_TIME	Converted time in HOST TIME. Format is HHMMSS.	int

Convert Date and Time Example Message

```
 <methodName>picture.convertDateTime</methodName>
  <params>
   <param>
    <value>0VCFZRFNJN</value>
    <value>20041215</value>
    <value>123000</value>
    <value>5</value>
   </param>
  </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><struct>
   <member><name>CONVERTED_DATE</name><value><int>20041215</int></value></member>
   <member><name>CONVERTED_TIME</name><value><int>73000</int></value></member>
  </struct></value></param>
 </params>
</methodResponse>
```

Directory Listing (picture.directoryListing)

This request lists the contents of the specified directory on the host.

Table 70. Directory Listing Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
PATH	Fully qualified path for which the directory listing should be done.	string

Table 71. Directory Listing Output Parameters

Field Name	Field Description	Data Type
FILE	Name of a file located in the specified PATH. The Directory Listing request will return an array of "FILE" key-value parts, one for each file in the specified PATH.	string

Directory Listing Example Message

```
 <methodName>picture.directoryListing</methodName>
  <params>
   <param>
    <value>0VCFZRFNJN</value>
    <value><struct>
     <member><name>PATH</name><value><string>/cas/db/backup</string></value></member>
    </struct></value>
   </param>
  </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><array><data>
   <value><struct>
    <member><name>FILE</name><value><string>backup.1216</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>FILE</name><value><string>backup.1217</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>FILE</name><value><string>backup.1218</string></value></member>
   </struct></value>
   </data></array></value></param>
  </params>
</methodResponse>
```

Get Alarm Status (picture.getAlarmStatus)

This request retrieves the general alarm status information of the alarm buffer.

Table 72. Get Alarm Status Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 73. Get Alarm Status Output Parameters

Field Name	Field Description	Data Type
LAST_INPUT_ID	ID of the last alarm input.	int
LAST_ALARM_ID	ID of the last alarm.	int
NUMBER_ACTIVE	Total number of active alarms in the buffer.	int
NUMBER_PENDING	Total number of pending alarms in the buffer.	int

Get Alarm Status Example Message

```
 <methodName>picture.getAlarmStatus</methodName>
  <params>
   <param>
    <value>0VCFZRFNJN</value>
    </param>
  </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
    <member><name>LAST_INPUT_ID</name><value><int>31</int></value></member>
    <member><name>LAST_ALARM_ID</name><value><int>15</int></value></member>
    <member><name>NUMBER_ACTIVE</name><value><int>77</int></value></member>
    <member><name>NUMBER_PENDING</name><value><int>3</int></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

Get Configured Devices (picture.getConfiguredDevices)

This request displays the counts of configured devices on the system.

Table 74. Get Configured Devices Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 75. Get Configured Devices Output Parameters

Field Name	Field Description	Data Type
MICRO	Number of micros configured on the system.	int
PORT	Number of ports configured on the system.	int
OPERATOR	Number of operators configured on the system.	int
HOST	Number of hosts configured on the system.	int
TAPE_DRIVE	Number of tape drives configured on the system.	int
STATUS_REQUEST	Number of status requests on the system.	int
MODEM	Number of modems configured on the system.	int

Get Configured Devices Example Message

```
 <methodName>picture.getConfiguredDevices</methodName>
  <params>
   <param>
    <value>0VCFZRFNJN</value>
    </param>
  </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
    <member><name>MICRO</name><value><int>15</int></value></member>
    <member><name>PORT</name><value><int>17</int></value></member>
    <member><name>OPERATOR</name><value><int>32</int></value></member>
    <member><name>HOST</name><value><int>1</int></value></member>
    <member><name>TAPE_DRIVE</name><value><int>1</int></value></member>
    <member><name>STATUS_REQUEST</name><value><int>0</int></value></member>
    <member><name>MODEM</name><value><int>1</int></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

Get Diagnostics (picture.getDiagnostics)

This request displays the diagnostics that are currently enabled.

Table 76. Get Diagnostics Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 77. Get Diagnostics Output Parameters

Field Name	Field Description	Data Type
DIAGNOSTICS	Long decimal value that contains all of the bitwise diagnostics flags that are to be enabled or disabled. The bitwise diagnostic flags are as follows: mumgr=1073741824 eifreqmgr=536870912 eifevtmgr=268435456 eflash=134217728 ucs=67108864 evtmgr=33554432 keymgr=16777216 cfgmgr=8388608 nwmdrv=4194304 pcsrvd=2097152 tracer=1048576 tcmgr=524288 mrtmgr=262144 tourmgr=131072 timer=65536 log=32768 schmgr=16384 stsmgr=8192 mreset=4096 rsndmgr=2048 snddrv=1024 rcvdrv=512 moddrv=256 oprmgr=128 almmgr=64 bdgmgr=32 prmgr=16 dbmgr=8 sndmgr=4 rcvmgr=2 main=1	string
LEVEL	Current diagnostics level (0-9).	int

Get Diagnostics Example Message

```
</methodCall>
 <methodName>picture.getDiagnostics</methodName>
  <params>
   <param>
    <value>0VCFZRFNJN</value>
    </param>
  </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
    <member><name>DIAGNOSTICS</name><value><string>0</string></value></member>
    <member><name>LEVEL</name><value><int>0</int></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

Get History Counts (picture.getHistoryCounts)

This request displays the history counts.

Table 78. Get History Counts Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 79. Get History Counts Output Parameters

Field Name	Field Description	Data Type
ALARM_HISTORY	Total number of alarm_history records.	int
BADGE_HISTORY	Total number of badge_history records.	int
EVENT_HISTORY	Total number of event_history records.	int
OPERATOR_HISTORY	Total number of operator_history records.	int
TOUR_HISTORY	Total number of tour_history records.	int
SYSTEM_HISTORY	Total number of system_history records.	int

Get History Counts Example Message

```
<methodCall>
 <methodName>picture.getHistoryCounts</methodName>
  <params>
   <param>
    <value>0VCFZRFNJN</value>
    </param>
  </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
    <member><name>ALARM_HISTORY</name><value><int>1203</int></value></member>
    <member><name>BADGE_HISTORY</name><value><int>52823</int></value></member>
    <member><name>EVENT_HISTORY</name><value><int>0</int></value></member>
    <member><name>OPERATOR_HISTORY</name><value><int>892</int></value></member>
    <member><name>TOUR_HISTORY</name><value><int>0</int></value></member>
    <member><name>SYSTEM_HISTORY</name><value><int>7628</int></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

Get History Flags (picture.getHistoryFlags)

This request displays the history time to archive flags.

Table 80. Get History Counts Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 81. Get History Counts Output Parameters

Field Name	Field Description	Data Type
ALARM_HISTORY	State indicating if it is time to archive alarm_history. Must be one of the following: YES or NO	string
BADGE_HISTORY	State indicating if it is time to archive badge_history. Must be one of the following: YES or NO	string
EVENT_HISTORY	State indicating if it is time to archive event_history records. Must be one of the following: YES or NO	string
OPERATOR_HISTORY	State indicating if it is time to archive operator_history. Must be one of the following: YES or NO	string
TOUR_HISTORY	State indicating if it is time to archive tour_history. Must be one of the following: YES or NO	string

Get History Flags Example Message

```
<methodCall>
 <methodName>picture.getHistoryFlags</methodName>
  <params>
   <param>
    <value>0VCFZRFNJN</value>
    </param>
  </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct>
    <member><name>ALARM_HISTORY</name><value><string>NO</string></value></member>
    <member><name>BADGE_HISTORY</name><value><string>NO</string></value></member>
    <member><name>EVENT_HISTORY</name><value><string>NO</string></value></member>
    <member><name>OPERATOR_HISTORY</name><value><string>NO</string></value></member>
    <member><name>TOUR_HISTORY</name><value><string>NO</string></value></member>
   </struct></value>
  </param>
 </params>
</methodResponse>
```

Get Host ID (picture.getHostId)

This request displays the host ID.

Table 82. Get Host ID Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 83. Get Host ID Output Parameters

Field Name	Field Description	Data Type
HOST_ID	Database host ID of the server.	int

Get Host ID Example Message

```
<methodCall>
 <methodName>picture.getHostId</methodName>
  <params>
   <param>
    <value>0VCFZRFNJN</value>
   </param>
  </params>
</methodCall>

<methodResponse>
 <params>
  <param>
   <value><struct><member><name>HOST_ID</name><value><boolean>1</boolean></value></member> </struct></value>
  </param>
 </params>
</methodResponse>
```

Get License Information (picture.getLicenseInformation)

This request retrieves the license information for all licensed packages.

Table 84. Get License Information Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 85. Get License Information Output Parameters

Field Name	Field Description	Data Type
REASON	State of the BASE license key, which must be one of the following: VALID "A fatal error occurred attempting to verify your key!" "Unable to access your key!" "Sorry, you have an invalid key!"	int
EXPIRES	Indicates if your key has an expiration date. Must be one of the following: YES or NO	string
EXPIRATION_YEAR	(only applicable if EXPIRES=YES) Expiration year	int
EXPIRATION_MONTH	(only applicable if EXPIRES=YES) Expiration month.	int
EXPIRATION_DAY	(only applicable if EXPIRES=YES) Expiration day.	int
READER_LIMIT	Maximum reader capacity.	int
TERMINAL_LIMIT	Maximum concurrent operator capacity.	int
IMAGE_TERMINAL_LIMIT	Maximum concurrent imaging workstation capacity.	int
CUSTOMER_CODE	Customer code by which licensing is applied.	string
LICENSE_KEY	License key currently installed.	string
PRODUCT_LIST	Array of PRODUCT_NAME key-value pairs.	array

Table 85. Get License Information Output Parameters (continued)

Field Name	Field Description	Data Type
PRODUCT_NAME	Product that is licensed. Multiple PRODUCT_NAME key-value pairs can exist in an array, one per each licensable product. The valid product names are as follows: ALPHACOM AMERICAN_DYNAMICS BASE BURLE_ALLEGIANT BURLE_ALLPLEX CENTRAL_STATION COMMEND DMP_SCS EDICON EIF EST FIRESINE GRUNDIG GUARD_TOURS IMAGING IMPORT_EXPORT JAVELIN KALATEL MAXPRO NETLAN NOTIFIER OSBOURNE_HOFFMAN_RECEIVER PANASONIC_550 PANASONIC_PROTEUS PELCO/PELCOB PPRS RAN SIEBE_CBAS SIMPLEX STENTOFON_9600 STENTOFON SUBHOST UCS UNIVERSAL_INTRUSION VICON_VIPER VISITOR	array
RESULT	Success or failure of this request for licensing information.	boolean

Get License Information Example Message

```
<methodCall>
 <methodName>picture.getLicenseInformation</methodName>
  <params>
   <param>
    <value>0VCFZRFNJN</value>
   </param>
  </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><struct>
   <member><name>REASON</name><value><string>VALID</string></value></member>
   <member><name>EXPIRES</name><value><string>NO</string></value></member>
   <member><name>READER_LIMIT</name><value><int>256</int></value></member>
   <member><name>TERMINAL_LIMIT</name><value><int>10</int></value></member>
   <member><name>IMAGE_TERMINAL_LIMIT</name><value><int>0</int></value></member>
   <member><name>CUSTOMER_CODE</name><value><string>593EGA3EJD</string></value></member>
   <member><name>LICENSE_KEY</name><value><string>S13H16FK-UFEXVFFC-1F78SQFS-
    XFQVFBUH-FCEPF</string></value></member>
   <member><name>PRODUCT_LIST</name><value>
    <array>data
     <value><struct><member>
      <name>PRODUCT_NAME</name><value><string>BASE</string></value></member>
     </struct></value>
     <value><struct><member>
      <name>PRODUCT_NAME</name><value><string>EIF</string></value></member>
     </struct></value>
     <value><struct><member>
      <name>PRODUCT_NAME</name><value><string>IMAGING</string></value></member>
     </struct></value>
    <data><array>
   <value><member>
   <member><name>RESULT</name><value><boolean>1</boolean></value></member>
  <value><struct></param>
 </params>
</methodResponse>
```

Get Micro Time (picture.microTime)

This request displays the date and time for a micro in the specified time zone.

Table 86. Get Micro Time Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
N/A	ID of a time zone.	int

Table 87. Get Micro Time Output Parameters

Field Name	Field Description	Data Type
MICRO_DATE	Current date in the specified time zone. Format is: YYYYMMDD.	int
MICRO_TIME	Current time in the specified time zone. Format is: HHMMSS.	int

Get Micro Time Example Message

```
<methodCall>
 <methodName>picture.microTime</methodName>
  <params>
   <param>
    <value><string>0VCFZRFNJN</string></value>
    <value><int>-1</int></value>
   </param>
  </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><struct>
   <member><name>MICRO_DATE</name><value><int>20041125</int></value></member>
   <member><name>MICRO_TIME</name><value><int>123015</int></value></member>
  </struct></value></param>
 </params>
</methodResponse>
```

Get Operators Logged In (picture.getOperatorsLoggedIn)

This request displays the session details for each operator currently logged on.

Table 88. Get Operators Logged In Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 89. Get Operators Logged In Output Parameters

Field Name	Field Description	Data Type
SESSION	Session ID used by the operator's login.	string
USER_ID	User ID of the operator.	string
ADDRESS	IP address of the operator's client.	string
PORT	Port number of the operator's client.	int
ID	ID of the operator record.	int
TIMEZONE	Default time zone of the operator record.	int
LOGIN_DATE	Date the operator most recently logged on. Format is in YYYYMMDD.	int
LOGIN_TIME	Time the operator most recently logged on. Format is in HHMMSS.	int

Get Operators Logged In Example Message

```
<methodCall>
 <methodName>picture.getOperatorsLoggedIn</methodName>
  <params>
   <param>
    <value>0VCFZRFNJN</value>
   </param>
  </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><array><data>
   <value><struct>
   <member><name>SESSION</name><value><string>JH3SJ93801</string></value></member>
   <member><name>USER_ID</name><value><string>install</string></value></member>
    <member><name>ADDRESS</name><value><string>192.168.2.10</string></value></member>
    <member><name>PORT</name><value><int>8085</int></value></member>
    <member><name>ID</name><value><int>1</int></value></member>
    <member><name>TIMEZONE</name><value><int>-1</int></value></member>
    <member><name>LOGIN_DATE</name><value><int>20041215</int></value></member>
    <member><name>LOGIN_TIME</name><value><int>123015</int></value></member>
   </struct></value>
   <value><struct>
    <member><name>SESSION</name><value><string>AN28DJ1083</string></value></member>
    <member><name>USER_ID</name><value><string>ehartman</string></value></member>
    <member><name>ADDRESS</name><value><string>192.168.2.155</string></value></member>
    <member><name>PORT</name><value><int>8087</int></value></member>
    <member><name>ID</name><value><int>5</int></value></member>
    <member><name>TIMEZONE</name><value><int>-1</int></value></member>
    <member><name>LOGIN_DATE</name><value><int>20041215</int></value></member>
    <member><name>LOGIN_TIME</name><value><int>101257</int></value></member>
   </struct></value>
  </data></array></value></param>
 </params>
</methodResponse>
```

Get Host Time (picture.hostTime)

This request displays the host date and time.

Table 90. Get Host Time Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 91. Get Host Time Output Parameters

Field Name	Field Description	Data Type
HOST_DATE	Current date in HOST TIME. Format is: YYYYMMDD.	int
HOST_TIME	Current time in HOST TIME. Format is: HHMMSS.	int

Get Host Time Example Message

```
<methodCall>
 <methodName>picture.hostTime</methodName>
  <params>
   <param>
    <value>0VCFZRFNJN</value>
   </param>
  </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><struct>
   <member><name>HOST_DATE</name><value><int>20041125</int></value></member>
   <member><name>HOST_TIME</name><value><int>123015</int></value></member>
  </struct></value></param>
 </params>
</methodResponse>
```

Login (picture.loginEIF)

This request authenticates a user ID and password and creates a valid session based on the SESSION ID specified. If successful, subsequent EIF Events will be transmitted to the ADDRESS and PORT specified.

Table 92. Login Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID to be assigned to the user ID if login is successful.	string
USER_ID	User ID .	string
PASSWORD	"SHA-1" digested password. This must match identically to the password of the operator record in the database.	string
ADDRESS	IP address of the client that is logging in.	string
PORT	Port number from which the client that is logging in will receive EIF Events from the Picture Perfect host.	int
UTC_TIME	(Used for logging operator history) Date and Time in UTC TIME. Format: YYYYMMDDHHMMSS Example: 20041215113000=December 15th, 2004 11:30AM	string
OPERATOR_TIME	(Used for logging operator history) Date and Time in OPERATOR TIME. Format: YYYYMMDDHHMMSS Example: 20041215113000=December 15th, 2004 11:30AM	string

Table 93. Login Output Parameters

Field Name	Field Description	Data Type
RESULT	True if the login was successful.	boolean
REASON	Reason for the login's RESULT, which must be one of the following: VALID=Credentials are valid INVALID_ACCOUNT=Credentials are invalid OPERATOR_LIMIT=No operator license available CAS_FULL=The /**cas** filesystem is full TMP-FULL=The /**tmp** filesystem is full POPEN_FAILED=A popen failure has occurred UKNOWN=An unknown error has occurred	string
DBHOST	The Informix database hostname	string
DBPORT	Informix database port number that clients should use for making database connections.	int
DBCSM	Informix database encryption protocol information. (Optional, this is only populated if database encryption is enabled.)	string
DBPASS	Informix database password that clients should use for making database connections.	string

Table 93. Login Output Parameters (continued)

Field Name	Field Description	Data Type
TERM_ID	Imaging terminal ID (terminal record) that corresponds to this client according to the IP address provided using the ADDRESS parameter.	int
HOST_TYPE	Type of Picture Perfect system the client is connecting to, which will be one of the following: 1=STANDALONE 2=PRIMARY 3=BACKUP 4=NETHOST 5=SUBHOST	int
NET_HOST_NAME	Hostname of the network host. (Optional, this is only populated if connecting to a subhost in a Picture Perfect Enterprise System.)	string
IMAGING_SERVER	Hostname of the imaging server. (Optional, this is only populated if connecting to a host that uses a remote imaging server to provide its image database.)	string
RELEASE_NUMBER	Picture Perfect host's release level.	int
BUILD_NUMBER	Picture Perfect host's build level.	int
BUILD_DATE	Picture Perfect host's buld date.	string
INSTALL_OS	Picture Perfect host's operating system type.	string
PACKAGE_COUNT	Number of Picture Perfect packages installed.	int
PACKAGE_LIST	Array of objects that contain the PACKAGE_NAME and PATCH_LEVEL pairs of information for every Picture Perfect package installed.	array
PACKAGE_NAME	Name of a Picture Perfect package that is installed.	string
PATCH_LEVEL	Patch level of a Picture Perfect package that is installed.	int

Login Example Message

```
<methodCall>
 <methodName>picture.loginEIF</methodName>
 <params>
  <param><value>F30350FE1B</value></param>
  <param><value><struct>
   <member><name>USER_ID</name><value><string>install</string></value></member>
   <member><name>PASSWORD</name><value><string>313afa5189c150b7b0f3e6d39e0fa223f88ec42b
       </string></value></member>
   <member><name>ADDRESS</name><value><string>192.168.2.10</string></value></member>
   <member><name>PORT</name><value><int>8085</int></value></member>
   <member><name>UTC_TIME</name><value><string>20041215113000</string></value></member>
   <member><name>OPERATOR_TIME</name><value><string>20041215113000</string></value>
       </member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><struct>
   <member><name>RESULT</name><value><boolean>1</boolean></value></member>
   <member><name>REASON</name><value><string>VALID</string></value></member>
```

```
   <member><name>/DBHOST</name><value><string>pphost</string></value></member>
   <member><name>/DBPORT</name><value><int>1960</int></value></member>
   <member><name>/DBHOST</name><value><string>pphost</string></value></member>
   <member><name>/DBCSM</name><value><string></string></value></member>
   <member><name>/DBPASS</name><value><string>ximrofni</string></value></member>
   <member><name>/TERM_ID</name><value><int>-3</int></value></member>
   <member><name>/HOST_TYPE</name><value><int>1</int></value></member>
   <member><name>/NET_HOST_NAME</name><value><string>NETHOSTNAME</string></value></member>
   <member><name>/IMAGING_SERVER</name><value><string>SERVER</string></value></member>

   <member><name>/RELEASE_NUMBER</name><value><int>3</int></value></member>
   <member><name>/BUILD_NUMBER</name><value><int>232</int></value></member>
   <member><name>/BUILD_DATE</name><value><string>12/08/2004</string></value></member>
   <member><name>/INSTALL_OS</name><value><string>Linux</string></value></member>
   <member><name>/PACKAGE_COUNT</name><value><int>1</int></value></member>
   <member><name>/PACKAGE_LIST</name><value>
        <array><data>
         </value><struct><member>
          <name>/PACKAGE_NAME</name><value><string>base</string></value></member>
          <name>/PATCH_LEVEL</name><value><int>0</int></value></member>
         </struct></value>
        </data></array>
   </value></member>
  </struct></value></param>
 </params>
</methodResponse>
```

Query Access Points (picture.queryAccessPoints)

This request is a query for all access points on the host.

Table 94. Query Access Points Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	session ID used from the login.	string

Table 95. Query Access Points Output Parameters

Field Name	Field Description	Data Type
DOOR_ID	Door record ID.	int
DOOR_DESC	Door record description.	string
SENSOR_ID	Door input sensor record ID.	int
REX_ID	Door exit request input ID.	int
STRIKE_ID	Door strike output ID.	int
READER_ID	Reader ID associated with door.	int
DOOR_FORCED_INGRP_ID	Door forced input group record ID.	int
DOOR_HELD_INGRP_ID	Door held input group record ID.	int
PREALARM_INGRP_ID	Door prealarm input group record ID.	int
INVALID_BADGE_INGRP_ID	Door area input group record ID.	int
SUSPENDED_BADGE_INGRP_ID	Door area input group record ID.	int
LOST_BADGE_INGRP_ID	Door area input group record ID.	int
UNKNOWN_BADGE_INGRP_ID	Door area input group record.	int
FACILITY_ID	Door facility record ID.	int
FACILITY_DESC	Door facility record description.	string

Query Access Points Example Message

```
<methodCall>
 <methodName>picture.queryAccessPoints</methodName>
 <params>
  <param><value>4X76LAAK3I</value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><array><data>
   <value><struct>
    <member><name>DOOR_ID</name><value><int>1</int></value></member>
    <member><name>DOOR_DESC</name><value><string>100100 BOCA LOBBY EAST..</string></value></member>
    <member><name>SENSOR_ID</name><value><int>1</int></value></member>
    <member><name>REX_ID</name><value><int>2</int></value></member>
    <member><name>STRIKE_ID</name><value><int>1</int></value></member>
    <member><name>READER_ID</name><value><int>1</int></value></member>
    <member><name>DOOR_FORCED_INGRP_ID</name><value><int>27</int></value></member>
    <member><name>DOOR_HELD_INGRP_ID</name><value><int>26</int></value></member>
    <member><name>PREALARM_INGRP_ID</name><value><int>0</int></value></member>
```

```
    <member><name>INVALID_BADGE_INGRP_ID</name><value><int>19</int></value></member>
    <member><name>LOST_BADGE_INGRP_ID</name><value><int>20</int></value></member>
    <member><name>SUSPENDED_BADGE_INGRP_ID</name><value><int>21</int></value></member>
    <member><name>UNKNOWN_BADGE_INGRP_ID</name><value><int>22</int></value></member>
    <member><name>FACILITY_ID</name><value><int>1</int></value></member>
    <member><name>FACILITY_DESC</name><value><string>BOCA RATON</string></value></member>
   </struct></value>
  </data></array></value></param>
 </params>
</methodResponse>
```

Query Alarm Instructions (picture.queryAlarmInstructions)

This request is a query for all alarm instructions for a particular alarm on the host.

Table 96. Query Alarm Instructions Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
ALARM_ID	Alarm record ID.	int

Table 97. Query Alarm Intructions Output Parameters

Field Name	Field Description	Data Type
INSTRUCTION	Array of alarm instructions.	string

Query Alarm Instructions Example Message

```
<methodCall>
 <methodName>picture.queryAlarmInstructions</methodName>
 <params>
  <param><value>WRF0Z188NQ</value></param>
  <param><value><struct>
   <member><name>ALARM_ID</name><value><int>7</int></value></member>
  </struct></value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><array><data>
   <value><struct>
    <member><name>INSTRUCTION</name><value><string>VIEW CAMERA</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>INSTRUCTION</name><value><string>CALL 911</string></value></member>
   </struct></value>
  </data></array></value></param>
 </params>
</methodResponse>
```

Query Alarm Responses (picture.queryAlarmResponses)

This request is a query for all alarm responses on the host.

Table 98. Query Alarm Responses Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 99. Query Alarm Responses Output Parameters

Field Name	Field Description	Data Type
RESPONSE	Array of alarm responses.	string

Query Alarm Responses Example Message

```
<methodCall>
<methodName>picture.queryAlarmResponses</methodName>
 <params>
  <param><value>ICYCONJ8FS</value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><array><data>
   <value><struct>
    <member><name>RESPONSE</name><value><string>AREA FLOODED</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>RESPONSE</name><value><string>AREA IS FLOODED</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>RESPONSE</name><value><string>AREA IS SECURE</string></value></member>
   </struct></value>
  </data></array></value></param>
 </params>
</methodResponse>
```

Query Facilities (picture.queryFacilities)

This request is a query for all facilities on the host.

Table 100. Query Facilities Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 101. Query Facilties Output Parameters

Field Name	Field Description	Data Type
FACILITY_ID	Facility record ID.	int
FACILITY_DESC	Facility record description.	string

Query Facilities Example Message

```
<methodCall>
 <methodName>picture.queryFacilities</methodName>
 <params>
  <param><value>UC7QD2S9Q7</value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><array><data>
   <value><struct>
    <member><name>FACILITY_ID</name><value><int>1</int></value></member>
    <member><name>FACILITY_DESC</name><value><string>BOCA RATON</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>FACILITY_ID</name><value><int>5</int></value></member>
    <member><name>FACILITY_DESC</name><value><string>CHICAGO</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>FACILITY_ID</name><value><int>3</int></value></member>
    <member><name>FACILITY_DESC</name><value><string>DALLAS</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>FACILITY_ID</name><value><int>-1</int></value></member>
    <member><name>FACILITY_DESC</name><value><string>GLOBAL</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>FACILITY_ID</name><value><int>4</int></value></member>
    <member><name>FACILITY_DESC</name><value><string>LOS ANGELES</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>FACILITY_ID</name><value><int>2</int></value></member>
    <member><name>FACILITY_DESC</name><value><string>NEW YORK</string></value></member>
   </struct></value>
  </data></array></value></param>
 </params>
</methodResponse>
```

Query Input Groups (picture.queryInputGroups)

This request is a query for all input groups on the host.

Table 102. Query Input Groups Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 103. Query Input Groups Output Parameters

Field Name	Field Description	Data Type
INGRP_ID	Input group record ID.	int
INGRP_DESC	Input group record description.	string
FACILITY_ID	Input group facility record ID.	int
FACILITY_DESC	Input group facility record description.	int

Query Input Groups Example Message

```
<methodCall>
 <methodName>picture.queryInputGroups</methodName>
 <params>
  <param><value>4X76LAAK3I</value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><array><data>
   <value><struct>
    <member><name>INGRP_ID</name><value><int>1</int></value></member>
    <member><name>INGRP_DESC</name><value><string>CHECK /cas/log/chkfs.log</string></value></member>
    <member><name>FACILITY_ID</name><value><int>-1</int></value></member>
    <member><name>FACILITY_DESC</name><value><string>GLOBAL</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>INGRP_ID</name><value><int>2</int></value></member>
    <member><name>INGRP_DESC</name><value><string>CHECK /cas/log/chkdbspace.log</string></value></member>
    <member><name>FACILITY_ID</name><value><int>1</int></value></member>
    <member><name>FACILITY_DESC</name><value><string>BOCA RATON</string></value></member>
   </struct></value>
  </data></array></value></param>
 </params>
</methodResponse>
```

Query Inputs (picture.queryInputs)

This request is a query for all inputs on the host.

Table 104. Query Inputs Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 105. Query Inputs Output Parameters

Field Name	Field Description	Data Type
INPUT_ID	Input record ID.	int
INPUT_DESC	Input record description.	string
FACILITY_ID	Input facility record ID.	int
FACILITY_DESC	Input facility record description.	int

Query Inputs Example Message

```
<methodCall>
 <methodName>picture.queryInputs</methodName>
 <params>
  <param><value>4X76LAAK3I</value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><array><data>
   <value><struct>
    <member><name>INPUT_ID</name><value><int>1795</int></value></member>
    <member><name>INPUT_DESC</name><value><string>169-1-16 INPUT 1</string></value></member>
    <member><name>FACILITY_ID</name><value><int>-1</int></value></member>
    <member><name>FACILITY_DESC</name><value><string>GLOBAL</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>INPUT_ID</name><value><int>1796</int></value></member>
    <member><name>INPUT_DESC</name><value><string>169-1-17 INPUT 2</string></value></member>
    <member><name>FACILITY_ID</name><value><int>1</int></value></member>
    <member><name>FACILITY_DESC</name><value><string>BOCA RATON</string></value></member>
   </struct></value>
  </data></array></value></param>
 </params>
</methodResponse>
```

Query Micro State (picture.queryMicroState)

This request is a query for the state of the specified micro.

Table 106. Query Micro State Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
MICRO_ID	ID of the micro whose state should be queried. If you specify "-1", then the state of all micros will be returned.	int

Table 107. Query Micro State Output Parameters

Field Name	Field Description	Data Type
ONLINE	Online state of the micro, which will be one of the following: YES NO PENDING ONLINE PENDING OFFLINE	string
TRACE	Trace state of the micro, which will be one of the following: YES or NO	string
PRIMARY_CHANNEL	Primary channel state of the micro, which will be one of the following: YES or NO	string
SECONDARY_CHANNEL	Secondary channel state of the micro, which will be one of the following: YES or NO	string
ALARM	Slarm state of the micro, which will be one of the following: YES or NO	string
RESETTING	Reset state of the micro, which will be one of the following: YES or NO	string
ERROR	Error state of the micro, which will be one of the following: YES or NO	string
ACK_PENDING	Ack pending state of the micro, which will be one of the following: YES or NO	string
DIAL_NOW	Dial now state of the micro will be one of the following: YES or NO * = state not yet initialized	string
DIALING	Dialing state of the micro will be one of the following: YES or NO * = state not yet initialized	string
CONNECTED	Connection state of the micro, which will be one of the following: YES or NO	string
LAST_COMM_MONTH	Month of the last communication timestamp in the format MM.	int

Table 107. Query Micro State Output Parameters (continued)

Field Name	Field Description	Data Type
LAST_COMM_DAY	Day of the last communication timestamp in the format DD.	int
LAST_COMM_HOUR	Hour of the last communication timestamp in the format HH.	int
LAST_COMM_MINUTE	Minute of the last communication timestamp in the format MM.	int
LAST_COMM_SECOND	Second of the last communication timestamp in the format SS.	int
RESEND_COUNT	Number of packets on the resend list for the specified micro.	int
MICRO_ID	ID of the micro.	int
MICRO_ADDRESS	Address of the micro.	int
UPSTREAM	The ID of the upstream micro of this micro or one of the following: -1=NONE -2=HOST	int
DOWNSTREAM	ID of the downstream micro of this micro or one of the following: -1=NONE -2=HOST	int
TYPE	Type of micro, which will be one of the following: DIALUP NETWORK NETWORK_DIALUP DOWNSTREAM_DIALUP DIRECT_CONNECT	string
FLASH_STATUS	Flash status of the micro, which will be one of the following: INDETERMINATE SCHEDULED FLASHING_COMPLETE TERMINATED REQUIRE_OPERATOR REQUEST_CRC_MSG START_BLOCK_MSG DATA_BLOCK_MSG DOWNLOAD_COMPLETE_MSG MICRO_RESET RELOAD_BADGE	string
PACKETS_LEFT	Number of eFlash packets left to be transmitted for this micro to complete a flash.	int
PACKETS_QUEUED	Number of eFlash packets queued in total to complete a flash.	int

Query Micro State Example Message

```
<methodCall>
 <methodName>picture.queryMicroState</methodName>
 <params>
  <param>
   <value>OVCFZRFNJN</value>
   <value><struct>
    <member><name>MICRO_ID</name><value><int>1</int></value></member>
   </struct></value>
  <param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><array><data>
   <value><struct>
    <member><name>ONLINE</name><value><string>YES</string></value></member>
    <member><name>TRACE</name><value><string>NO</string></value></member>
    <member><name>PRIMARY_CHANNEL</name><value><string>YES</string></value></member>
    <member><name>SECONDARY_CHANNEL</name><value><string>NO</string></value></member>
    <member><name>ALARM</name><value><string>NO</string></value></member>
    <member><name>RESETTING</name><value><string>NO</string></value></member>
    <member><name>ERROR</name><value><string>NO</string></value></member>
    <member><name>ACK_PENDING</name><value><string>NO</string></value></member>
    <member><name>DIAL_NOW</name><value><string>*</string></value></member>
    <member><name>DIALING</name><value><string>*</string></value></member>
    <member><name>CONNECTED</name><value><string>YES</string></value></member>
    <member><name>LAST_COMM_MONTH</name><value><int>12</int></value></member>
    <member><name>LAST_COMM_DAY</name><value><int>15</int></value></member>
    <member><name>LAST_COMM_HOUR</name><value><int>9</int></value></member>
    <member><name>LAST_COMM_MINUTE</name><value><int>30</int></value></member>
    <member><name>LAST_COMM_SECOND</name><value><int>15</int></value></member>
    <member><name>RESENT_COUNT</name><value><int>0</int></value></member>
    <member><name>MICRO_ID</name><value><int>1</int></value></member>
    <member><name>MICRO_ADDRESS</name><value><int>1</int></value></member>
    <member><name>UPSTREAM</name><value><int>-2</int></value></member>
    <member><name>DOWNSTREAM</name><value><int>-1</int></value></member>
    <member><name>TYPE</name><value><string>NETWORK</string></value></member>
    <member><name>FLASH_STATUS</name><value><string>INDETERMINATE</string></value></member>
    <member><name>PACKETS_LEFT</name><value><int>0</int></value></member>
    <member><name>PACKETS_QUEUED</name><value><int>0</int></value></member>
   </struct></value>
  </data></array></value></param>
 </params>
</methodResponse> </params>
</methodResponse>
```

Query Modes (picture.queryModes)

This request is a query for all modes on the host.

Table 108. Query Modes Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 109. Query Modes Output Parameters

Field Name	Field Description	Data Type
MODE_ID	ID of the mode record.	int
MODE_DESC	Description of the mode record.	string

Query Modes Example Message

```
<methodCall>
 <methodName>picture.queryModes</methodName>
 <params>
  <param><value>ICYCONJ8FS</value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><array><data>
   <value><struct>
    <member><name>MODE_ID</name><value><int>1</int></value></member>
    <member><name>MODE_DESC</name><value><string>NORMAL</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>MODE_ID</name><value><int>2</int></value></member>
    <member><name>MODE_DESC</name><value><string>HOLIDAY</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>MODE_ID</name><value><int>3</int></value></member>
    <member><name>MODE_DESC</name><value><string>NORMAL</string></value></member>
   </struct></value>
  </data></array></value></param>
 </params>
</methodResponse>
```

Query Operators (picture.queryOperators)

This request is a query for all operators on the host.

Table 110. Query Operator Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 111. Query Operator Output Parameters

Field Name	Field Description	Data Type
USER_ID	Operator record ID.	int
NAME	Name of operator.	string
USERNAME	Operator login name.	string

Query Operator Example Message

```
<methodCall>
 <methodName>picture.queryOperators</methodName>
 <params>
  <param><value>AKE74UNUSQ</value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><array><data>
   <value><struct>
    <member><name>USER_ID</name><value><int>1</int></value></member>
    <member><name>NAME</name><value><string>system administrator</string></value></member>
    <member><name>USERNAME</name><value><string>install</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>USER_ID</name><value><int>4</int></value></member>
    <member><name>NAME</name><value><string>guard1</string></value></member>
    <member><name>USERNAME</name><value><string>guard1</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>USER_ID</name><value><int>5</int></value></member>
    <member><name>NAME</name><value><string>guard2</string></value></member>
    <member><name>USERNAME</name><value><string>guard2</string></value></member>
   </struct></value>
  </data></array></value></param>
 </params>
</methodResponse>
```

Query Outputs (picture.queryOutputs)

This request is a query for all outputs on the host.

Table 112. Query Outputs Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string

Table 113. Query Outputs Output Parameters

Field Name	Field Description	Data Type
OUTPUT_ID	Output record ID.	int
OUTPUT_DESC	Output record description.	string
FACILITY_ID	Output facility record ID.	int
FACILITY_DESC	Output facility record description.	string

Query Outputs Example Message

```
<methodCall>
 <methodName>picture.queryOutputs</methodName>
 <params>
  <param><value>4X76LAAK3I</value></param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><array><data>
   <value><struct>
    <member><name>OUTPUT_ID</name><value><int>1795</int></value></member>
    <member><name>OUTPUT_DESC</name><value><string>169-1-31 OUTPUT 1</string></value></member>
    <member><name>FACILITY_ID</name><value><int>-1</int></value></member>
    <member><name>FACILITY_DESC</name><value><string>GLOBAL</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>OUTPUT_ID</name><value><int>1796</int></value></member>
    <member><name>OUTPUT_DESC</name><value><string>169-1-32 OUTPUT 2</string></value></member>
    <member><name>FACILITY_ID</name><value><int>1</int></value></member>
    <member><name>FACILITY_DESC</name><value><string>BOCA RATON</string></value></member>
   </struct></value>
  </data></array></value></param>
 </params>
</methodResponse>
```

Query Port State (picture.queryPortState)

This request is a query for the state of the specified port.

Table 114. Query Port State Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
PORT_ID	ID of the port whose state should be queried. If you specify "-1", then the state of all ports will be returned.	int

Table 115. Query Port State Output Parameters

Field Name	Field Description	Data Type
PORT_ID	ID of the port.	int
PORT_NAME	Name of the port.	string
FILE_HANDLE	File handle of the port.	int
FIRST_MICRO_ID	ID of the first micro on this port.	int
MODEM_TYPE	Type of modem on this port.	int
SPEED	Baud rate of the port that can be one of the following: 300 1200 2400 4800 9600 19200 38400	int
PARITY	Parity of the port that can be one of the following: ODD, EVEN, NONE	string
DATA_BITS	Number of data bits of the port, which can be one of the following: 5, 6, 7, 8	int
STOP_BITS	Number of stop bits of the port, which can be one of the following: 1, 2	int
INITIALIZED	Initialization state of the port, which can be one of the following: YES, NO	string
SND	Send state of the port, which can be one of the following: YES, NO	string
RCV	Receive state of the port, which can be one of the following: YES, NO	string
DIALING	Dialing state of the port, which can be one of the following: YES, NO	string

Table 115. Query Port State Output Parameters

Field Name	Field Description	Data Type
CONNECTED	Connected state of the port, which can be one of the following: YES, NO	string

Query Port State Example Message

```
<methodCall>
 <methodName>picture.queryPortState</methodName>
 <params>
  <param>
   <value>OVCFZRFNJN</value>
   <value><struct>
    <member><name>PORT_ID</name><value><int>1</int></value></member>
   </struct></value>
  </param>
 </params>
</methodCall>

<methodResponse>
 <params>
  <param><value><array><data>
   <value><struct>
    <member><name>PORT_ID</name><value><int>1</int></value></member>
    <member><name>PORT_NAME</name><value><string>/dev/ttyD001</string></value></member>
    <member><name>FILE_HANDLE</name><value><int>65</int></value></member>
    <member><name>FIRST_MICRO_ID</name><value><int>3</int></value></member>
    <member><name>MODEM_TYPE</name><value><int>4</int></value></member>
    <member><name>SPEED</name><value><int>9600</int></value></member>
    <member><name>PARITY</name><value><string>ODD</string></value></member>
    <member><name>DATA_BITS</name><value><int>7</int></value></member>
    <member><name>STOP_BITS</name><value><int>1</int></value></member>
    <member><name>INITIALIZED</name><value><string>YES</string></value></member>
    <member><name>SND</name><value><string>NO</string></value></member>
    <member><name>RCV</name><value><string>NO</string></value></member>
    <member><name>DIALING</name><value><string>NO</string></value></member>
    <member><name>CONNECTED</name><value><string>YES</string></value></member>
   </struct></value>
  </data></array></value></param>
 </params>
</methodResponse>
```

Query Status (picture.queryStatus)

This request is a query for status information from micros.

Table 116. Query Status Input Parameters

Field Name	Field Description	Data Type
SESSION_ID	Session ID used from the login.	string
STATUS_FLAG	Type of status request to be performed. STATUS_BY_MICRO=0 STATUS_BY_AREA=1	int
ID	ID of the micro or area for which status is being requested.	int
TABLE_ID	ID of the table for which status is being requested.	int
BID	Badge encode number of the badge for which status is being requested.	string

Table 117. Query Status Output Parameters

Field Name	Field Description	Data Type
STATUS_LINE	Line of status information for the requested status.	string

Query Status Example Message

```
<methodCall>
 <methodName>picture.queryStatus</methodName>
 <params>
  <param>
   <value>OVCFZRFNJN</value>
   <value><struct>
    <member><name>STATUS_FLAG</name><value><int>0</int></value></member>
    <member><name>ID</name><value><int>8</int></value></member>
    <member><name>TABLE_ID</name><value><int>22</int></value></member>
    <member><name>BID</name><value><string></string></value></member>
   </struct></value>
  </param>
 </params>
</methodCall>

<methodResponse>
<params>
  <param><value><array><data>
   <value><struct>
    <member><name>STATUS_LINE</name><value><string>Application Part 5</string></value>
</member>
   </struct></value>
   <value><struct>
    <member><name>STATUS_LINE</name><value><string>Application Reserved 0</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>STATUS_LINE</name><value><string>Application Date 20041215</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>STATUS_LINE</name><value><string>Application Time 1706</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>STATUS_LINE</name><value><string>Application Number 299</string></value></member>
   </struct></value>
   <value><struct>
    <member><name>STATUS_LINE</name><value><string>OS Part 0</string></value></member>
   </struct></value>
  </data></array></value></param>
 </params>
</methodResponse>
```

Chapter 3 Examples

This section provides several examples of Picture Perfect EIF XML-RPC-compatible clients and servers in the Java and Perl programming languages.

In this chapter:

Overview 88
Java Implementation Examples 88
Perl Implementation Examples 93

Overview

The examples in this chapter successfully communicate with the Picture Perfect EIF Event and Request Managers using XML-RPC. You may use these examples as a starting point or guide when building your own Picture Perfect EIF clients/Servers.

When implementing an EIF client or Server in a particular programming language, it is likely that a third party XML-RPC package will need to be used to facilitate communications with the Picture Perfect EIF Event and Request managers using XML-RPC. The following examples use *XMLRPC* for Java and *Frontier-RPC* for Perl.

Java Implementation Examples

In order to execute the sample Java EIF client/Server, a third party XML-RPC package for Java must be accessible to your Java compiler. Please visit http://xml.apache.org for more information regarding XML-RPC for Java. Please visit http://xml.apache.org/dist/xmlrpc/release/v1.2-b1/ to obtain the `xmlrpc-1.2-b1.jar` file that is required to execute our examples.

Sample EIF XML-RPC Client in Java

The following sample EIF client logs in and creates a valid session to Picture Perfect EIF, queries for a list of all Picture Perfect operators, and logs out. It makes use of the EIF Commands `picture.loginEIF` and `picture.logout`. It also makes uses of the EIF Data Request `picture.queryOperators`. This example attempts to connect to an EIF Request Manager on a Picture Perfect host named `bcttux` using port 8088.

```
package client;

/**
 * Title:  EifClient.java
 * Description:  Sample EIF Client For Java
 * Copyright:    Copyright (c) 2003-2004
 * Company:  GE Security
 * @author
 * @version 1.0
 */

import java.util.Vector;
import java.util.Hashtable;
import java.lang.Object;
import java.lang.Exception;
import org.apache.xmlrpc.*;


public class EifClient {
    /**
     * Session ID to use when connecting with the host.
```

```
 */
 private final static String SESSION_ID = "F30350FE1B696988B0EF6E50EFB4F07F";

/**
 * Picture perfect host to connect to.  Port must match the value
 * specified in /cas/db/text/eifreqmgr.cfg.  Replace the hostname with
 * the appropriate hostname.
 */
private final static String SERVER_URL = "http://bcttux:8088/";

public static void main (String [] args) {
   try {
         // Create an object to represent our server.
         XmlRpcClient server = new XmlRpcClient(SERVER_URL);

         /*
          * Build a hash table containing arguments to the
          * login() RPC method about to be invoked on the host.
          */
         final Hashtable h = new Hashtable();
         h.put("USER_ID", "install");    /* Picture Perfect username */
         h.put("PASSWORD", "89fb511ffe93ee7826661cale3bb468dclad0ff2");   /* Password */
         h.put("ADDRESS", "192.168.2.10");   /* IP Address of client */
         h.put("PORT", new Integer (8085));   /* Port of client to receive EIF Events */
         h.put("UTC_TIME", "20041215113015");   /* Now in UTC TIME */
         h.put("OPERATOR_TIME", "20041215113015");   /* Now in OPERATOR TIME */

         /*
          * Build the parameter list containging a unique session
          * identifier and the argument block just constructed above.
          */
         Vector params = new Vector();
         params.addElement(SESSION_ID);
         params.addElement(h);

         // Invoke the loginEIF() method on the host.
         System.out.println("Executing loginEIF()...");
         Hashtable result =
          (Hashtable) server.execute("picture.loginEIF", params);

         // Fetch the result from the RCP result set.
         final Object loginResult = result.get("RESULT");

         if (loginResult != null && loginResult instanceof Boolean &&
          ((Boolean) loginResult).booleanValue()) {
          System.out.println("Successfully logged in...");

          /*
```

```
                 * Build the parameter list containing a unique session
                 * identifier only.
                 */
                params = new Vector();
                params.addElement(SESSION_ID);

                // Invoke the queryOperators() method on the host.
                System.out.println("Querying operators...");
                Vector result2 = (Vector) server.execute("picture.queryOperators",
                        params);

                // Print out our result.
                for (int n = 0; n < result2.size(); n++)
                   System.out.println("Operator[" + n + "]=" + result2.get(n));

                /*
                 * Build the parameter list containing a unique session
                 * identifier only.
                 */
                params = new Vector();
                params.addElement(SESSION_ID);
                final Hashtable h2 = new Hashtable();
                h2.put ("UTC_TIME", "20041215113015"); /* Now in UTC TIME */
                params.addElement (h2);

                // Invoke the logout() method on the host.
                System.out.println("Logging out...");
                result = (Hashtable) server.execute("picture.logout",
                   params);

                // Fetch the result from the RCP result set.
                final Object logoutResult = result.get("RESULT");

                if (logoutResult != null && logoutResult instanceof Boolean &&
                   ((Boolean) logoutResult).booleanValue())
                        System.out.println("Successfully logged out...");
                else
                        System.out.println("Logout unsuccessful...");
               } else
                System.out.println("Login unsuccessful...");
        } catch (XmlRpcException exception) {
              System.err.println("EifClient: XML-RPC Fault #" +
                      Integer.toString(exception.code) + ": " +
                      exception.toString());
        } catch (Exception exception) {
              System.err.println("EifClient: " + exception.toString());
        }
    }
}
```

Sample EIF XML-RPC Server in Java

The following sample EIF Server will accept incoming EIF Events from an EIF Event Manager and display the contents of each event that is received. Notice that the EifServer class provides methods that correlate to the EIF Event methods. Whenever this server receives an EIF Event, the corresponding method is invoked. In this example, the EIF Event Manager on the Picture Perfect host should be configured with a URL set to "http://<hostname>/" and PORT set to 8085. In this example, <hostname> is `bcttux`.

Note: The URL should refer to the hostname of the machine running the Java EIF Server, which may or may not necessarily be the Picture Perfect hostname.

```
package eifserver;
/**
 *                 Copyright (c) 2003-2004 GE Security
 *                         All Rights Reserved.
 *
 * @(#) EifServer 1.0 5/14/2003
 *
 *
 */

import java.util.Hashtable;
import java.util.Vector;
import org.apache.xmlrpc.*;


public class EifServer {
       /**
        * Canned result set for RPC.
        */
       private final static Hashtable RESULT = new Hashtable();

       /**
        * Name of the RPC object.  This should be the hostname of the
        * connecting Picture Perfect host serving the events.
        */
       private final static String RPC_OBJECT = "bcttux";

       /**
        * Port to run the service on.
        */
       private final static int PORT = 8085;

       /**
        * Static initializer for default RPC result.
        */
       static {
              RESULT.put("RESULT", Boolean.TRUE);
       }
```

```
        public Hashtable ping(Hashtable arguments) {
                return RESULT;
        }

        public Hashtable processAlarm(Hashtable arguments) {
                System.out.println("processAlarm(): " + arguments);
                return RESULT;
        }

        public Hashtable processAlarmProcessState(Hashtable arguments) {
                System.out.println("processAlarmProcessState(): " + arguments);
                return RESULT;
        }

        public Hashtable processAlarmRefresh(Hashtable arguments) {
                System.out.println("processAlarmRefresh(): " + arguments);
                return RESULT;
        }

        public Hashtable processBadge(Hashtable arguments) {
                System.out.println("processBadge(): " + arguments);
                return RESULT;
        }

        public static void main (String [] args) {
              try {
                   // Invoke me as <http://localhost:8080/RPC2>.
                   WebServer server = new WebServer(PORT);

                   /*
                    * Add a handle for the RPC_OBJECT.  The methods defined in
                    * this class are exposed to the host.
                    */
                   server.addHandler(RPC_OBJECT, new EifServer());

                   // Start the service.
                   server.start();
                } catch (Exception exception) {
                   System.err.println("EifServer: " + exception.toString());
                }
        }
}
```

Perl Implementation Examples

In order to execute the sample Perl EIF client/Server, a third party XML-RPC module for Perl must first be installed. Please visit *http://www.xmlrpc.com* for more information regarding XML-RPC for Perl. Please visit *http://search.cpan.org.dist/Frontier-RPC/* to obtain the Frontier-RPC Perl module that is required to execute our examples.

Sample EIF XML-RPC Client in Perl

The following sample EIF client logs in and creates a valid session to Picture Perfect EIF, queries for a list of all Picture Perfect operators, and logs out. It makes use of the EIF Commands `picture.loginEIF` and `picture.logout`. It also makes uses of the EIF Data Request `picture.queryOperators`. This example attempts to connect to an EIF Request Manager on a Picture Perfect host named `bcttux` using port 8088.

```
#!/usr/bin/perl
#
#  Copyright (c) 2003-2004 GE Security
#
#  eifclient.pl
#
#  Sample Picture Perfect EIF Client
#
#  Created 05/14/03
#
#

use Frontier::Client;

# Setup the hostname:port of the Picture Perfect host with EIF installed
$server_url = 'http://bcttux:8088';
$session_id = 'F30350FE1B696988B0EF6E50EFB4F07F';

# Setup user authentication information for a Picture Perfect Operator
$authentication_input = { 'USER_ID' => 'install', 'PASSWORD' => 'install' };

# Make an object to represent the XML-RPC server.
$server = Frontier::Client->new(url => $server_url, debug => 0);
# Call the remote server and get our result
$result = $server->call('picture.loginEIF', $session_id, $authentication_input);
$login_successful = $result->{'RESULT'}->value();
if ( $login_successful eq 0 )
{
       print "...Login Failed!\n";
} else {
       print "...Login Successful!\n";

       # Query for some operators
```

```
        $result = $server->call('picture.queryOperators', $session_id);

        $count = 0;

        # Print out every key-value pair stored in the result
        $my_count = @{$result};
        while ( $count ne $my_count )
        {
                  $pointer = @{$result}[$count];

                  foreach $i (keys %$pointer)
                  {
                        printf "%-25s: %s\n", $i, $$pointer{$i};
                  }
                  printf "\n";

                  $count = $count + 1;
  }

  # Logout of the remote server.
  $result = $server->call('picture.logout', $session_id);
  $logout_successful = $result->{'RESULT'}->value();

  if ( $logout_successful eq 0 )
  {
                  print "...Logout Failed!\n";
  } else {
                  print "...Logout Successful!\n";
  }
 } # end of Successful Login
```

Sample EIF XML-RPC Server in Perl

The following sample EIF Server will accept incoming EIF Events from an EIF Event Manager and display the contents of each event that is received. Notice that the $methods hash table contains a correlation between EIF Event methods and Perl subroutines. Whenever this server receives an EIF Event, the corresponding method's subroutine is invoked. In this example, the EIF Event Manager on the Picture Perfect host should be configured with a URL set to http://<hostname>/RPC2 and PORT set to 8085. In this example, <hostname> is `bcttux`.

Note: The URL should refer to the hostname of the machine running the Perl EIF Server, which may or may not necessarily be the Picture Perfect hostname.

```
#!/usr/bin/perl
#
#  Copyright (c) 2003-2004 GE Security
#
#  eifserver.pl
#
#  Sample Picture Perfect EIF Server
#
#  Created 05/14/03
#
#

use Frontier::Daemon;

# Functions
sub sub_process_badge {
       my ($my_hash) = @_;

       print "\n...Received Badge Event:\n";
       foreach $i (keys %$my_hash)
       {
              printf "%-25s: %s\n", $i, $$my_hash{$i};
       }

       return ('RESULT' => 1);
}

sub sub_process_alarm {
       my ($my_hash) = @_;

       print "\n...Received Alarm Event:\n";
       foreach $i (keys %$my_hash)
       {
              printf "%-25s: %s\n", $i, $$my_hash{$i};
       }

       return ('RESULT' => 1);
}
```

```
sub sub_process_ping {
	my ($my_hash) = @_;

	print "\n...Received Ping Event:\n";
	foreach $i (keys %$my_hash)
	{
		printf "%-25s: %s\n", $i, $$my_hash{$i};
	}

	return ('RESULT' => 1);
}

sub sub_process_alarm_process {
	my ($my_hash) = @_;

	print "\n...Received Alarm Process State Change Event:\n";
	foreach $i (keys %$my_hash)
	{
		printf "%-25s: %s\n", $i, $$my_hash{$i};
	}

	return ('RESULT' => 1);
}

sub sub_process_alarm_refresh {
	my ($my_hash) = @_;

	print "\n...Received Alarm Refresh Event:\n";
	foreach $i (keys %$my_hash)
	{
		printf "%-25s: %s\n", $i, $$my_hash{$i};
	}

	return ('RESULT' => 1);
}

$methods = { 'bcttux.processBadge' => \&sub_process_badge,
	     'bcttux.processAlarm' => \&sub_process_alarm,
	     'bcttux.ping' => \&sub_process_ping,
	     'bcttux.processAlarmProcessState' => \&sub_process_alarm_process,
	     'bcttux.processAlarmRefresh' => \&sub_process_alarm_refresh
	   };

new Frontier::Daemon( LocalPort => 8085, methods => $methods )
	or die "Couldn't start HTTP server: $!";
```

Chapter 4 Contacting technical support

This chapter provides information to contact technical support in case you need assistance with your GE equipment.

In this chapter:

Contacting technical support 98

Contacting technical support

For assistance installing, operating, maintaining, and troubleshooting this product, refer to this document and any other documentation provided. If you still have questions, you may contact technical support during normal business hours (Monday through Friday, 8 a.m. to 7 p.m. Eastern Time. Protection plans are available for extended coverage.

Table 118. Sales and support contact information

	Pre-sales	Technical support
Phone	800.428.2733	888.GESECURITY (888.437.3287 Outside the toll-free area: 503.885.5700.
E-mail	None	rs-bctsupport@ge.com
Fax	561.998.6160	561.998.6224 (available 24 hours a day)

Note: Be ready at the equipment before calling for technical support.

Online publication library

Another great resource for assistance with your GE product is our online publication library, available to all of our customers. To access the library, go to our website at the following location:

`http://www.gesecurity.com`

In the **Tools** area at the top, click the *Publication Library* link. After you register and log on, you may search through our online library for the documentation you need.[1]

1. Many GE documents are provided as PDFs (portable document format). To read these documents, you will need Adobe Acrobat Reader, which can be downloaded free from Adobe's website at `www.adobe.com`.

Index

A

Activate Output 32
Alarm Event Object 15
Alarm Events 3
Alarm Process State Change Event Object 17
Alarm Refresh Event Array Object 18
authentication 2

B

Badge Event Object 21
Badge Events 3

C

cas/db/text 4
Change Mode 33
Change Modes 41
Check or Add Port 34
client authentication 5
Commands 2
conventions vii
Convert Date and Time 54
Create Operator History 39

D

daemons 2, 3
Data Requests 2, 12
Directory Listing 55
Disable Input 35
Disable Input Group on Micro 35

E

EIF Client 3
EIF Commands 12, 30
EIF Events 12, 13
EIF Server 3
eifevtmgr.cfg 4, 5
eifreqmgr.cfg 4, 5
Enable Input 36
Enable Input Group 36
encryption 2
Event Response 14
Event Response Object 30
Events 2

F

Force Alarm Refresh 32

G

Get Alarm Status 56
Get Configured Devices 57
Get Diagnostics 58
Get History Counts 60
Get History Flags 61
Get Host ID 62
Get Host Time 68
Get License Information 63
Get Micro Time 66
Get Operators Logged In 67

H

HTTP 6
HTTPS 6

I

Informix 2
Input Event Object 24

L

Lock Door 37
Login 3, 69
Logout 3, 38

O

Operator Event Object 24

P

Ping Event Object 14
Popup Event Object 25
PPRS Mode Change Event Object 26
pprscmd start 4
pprscmd stop 4

preface vii
Process Alarm 52
publication library 98

Q

Query Access Points 72
Query Alarm Instructions 73
Query Alarm Responses 74
Query Facilities 3, 75
Query Input Groups 76
Query Inputs 77
Query Micro State 78
Query Modes 81
Query Operators 3, 82
Query Outputs 83
Query Port State 84
Query Status 86

R

rc.pperf 4
rc.pperf –k 4
remote clients 2
Request Protocol 54
Reset Micro 43
Reset Output 44
Reset Timed APB 45

S

safety terms and symbols vii
Set Diagnostics 46
Set Ping Interval 48
Single Alarm Refresh Event Object 26
socket connection 5, 6
SSL 2

T

technical support 98
Tour Event Object 29
Tour Functions 49
TPS 3
tps_daemons 3
TTY Validate 50

U

Unlock Door 51

V

Validate Printer Queue 42

X

XML message 5
XML-RPC 2

Customer Survey

Title of this book: *Picture Perfect 4.0 External Interface User Manual*

My overall rating of this book:

- ☐ Excellent
- ☐ Very good
- ☐ Good
- ☐ Satisfactory
- ☐ Fair
- ☐ Poor

What I liked most about this book:

What I would change, add, delete, etc. in future editions of this book:

Please return this form to:

Attn: Technical Writing
GE Security
791 Park of Commerce Blvd. Suite 100
Boca Raton, FL 33487

or e-mail your comments to:
bctdocs@ge.com